

CD

with PLL  
design and  
simulation  
software



# Phase-Locked Loops

Sixth Edition

Design, Simulation,  
and Applications

Roland E. Best

# Introduction to PLLs

## Operating Principles of the PLL

The phase-locked loop (PLL) helps keep parts of our world orderly. If we turn on a television set, a PLL keeps heads at the top of the screen and feet at the bottom. In color television, another PLL makes sure green remains green and red remains red (even if politicians claim the reverse is true).

A PLL is a circuit that causes a particular system to track with another one. More precisely, a PLL is a circuit synchronizing an output signal (generated by an oscillator) with a reference or input signal in frequency as well as in phase. In the synchronized—often called “locked” —state, the phase error between the oscillator’ s output signal and the reference signal is zero, or it remains constant.

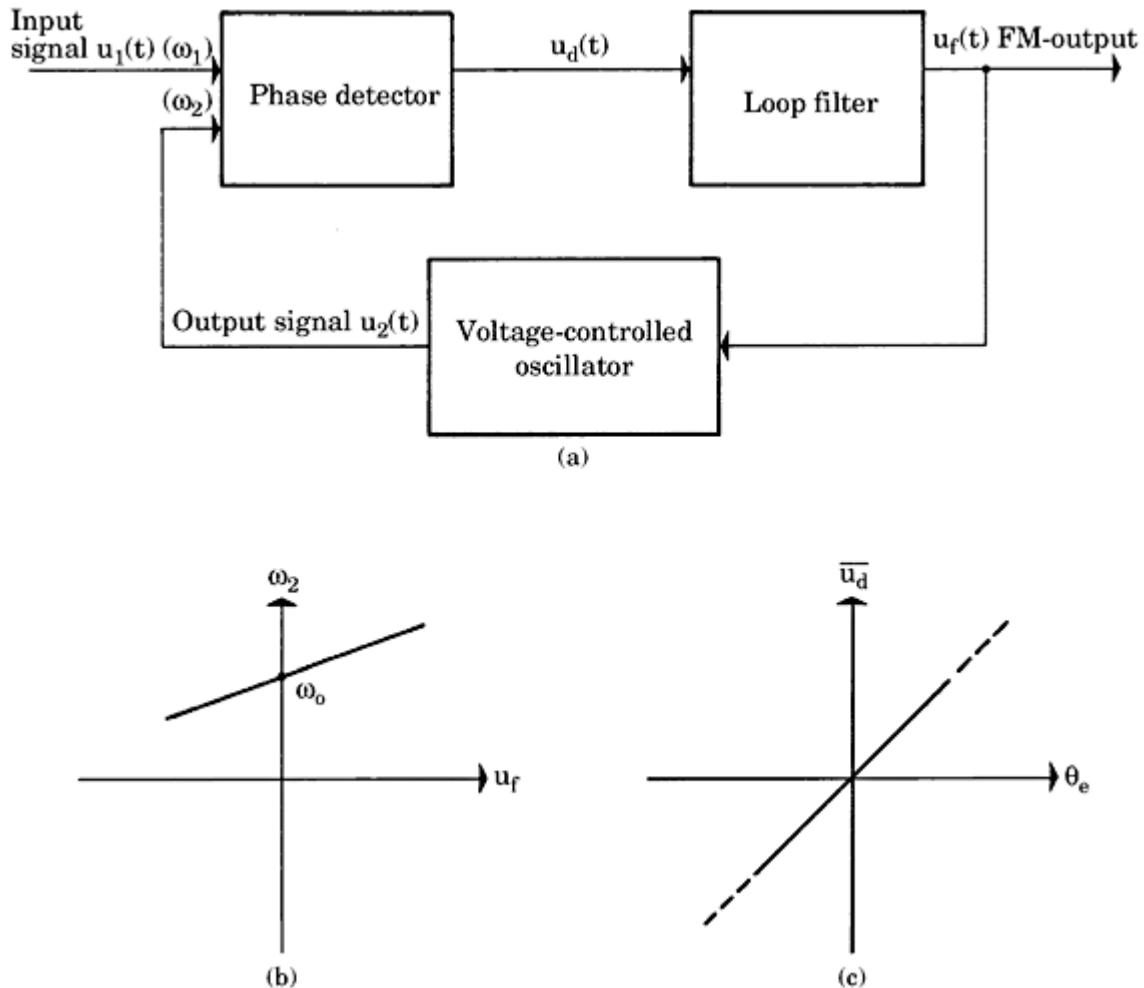
If a phase error builds up, a control mechanism acts on the oscillator in such a way that the phase error is again reduced to a minimum. In such a control system, the phase of the output signal is actually *locked* to the phase of the reference signal. This is why it is referred to as a *phase-locked loop*.

The operating principle of the PLL is explained by the example of the linear PLL (LPLL). As will be pointed out in [Sec. 1.3](#), other types of PLLs exist—for example, digital PLLs (DPLLs), all-digital PLLs (ADPLLs), and software PLLs (SPLLS). The PLL block diagram is shown in [Fig. 1.1a](#) and consists of three basic functional blocks:

- A voltage-controlled oscillator (VCO)
- A phase detector (PD)
- A loop filter (LF)

In this simple example, there is no down scaler between the output of VCO [ $u_2(t)$ ] and the lower input of the phase detector [ $\omega_2$ ]. Systems using down scalers are discussed in the following chapters.

In some PLL circuits, a current-controlled oscillator (CCO) is used instead of the VCO. In this case, the output signal of the phase detector is a controlled



**Figure 1.1** (a) Block diagram of the PLL. (b) Transfer function of the VCO. ( $u_f$  = control voltage;  $\omega_2$  = angular frequency of the output signal.) (c) Transfer function of the PD. ( $\bar{u}_d$  = average value of the phase-detector output signal;  $\theta_e$  = phase error.)

current source rather than a voltage source. However, the operating principle remains the same. The signals of interest within the PLL circuit are defined as follows:

- The reference (or input) signal  $u_1(t)$
- The angular frequency  $\omega_1$  of the reference signal
- The output signal  $u_2(t)$  of the VCO
- The angular frequency  $\omega_2$  of the output signal
- The output signal  $u_d(t)$  of the phase detector
- The output signal  $u_f(t)$  of the loop filter
- The phase error  $\theta_e$ , defined as the phase difference between signals  $u_1(t)$  and  $u_2(t)$

**Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.**

Let us now look at the operation of the three functional blocks in [Fig. 1.1a](#). The VCO oscillates at an angular frequency  $\omega_2$ , which is determined by the output signal  $u_f$  of the loop filter. The angular frequency  $\omega_2$  is given by

$$\omega_2(t) = \omega_0 + K_0 u_f(t) \quad (1.1)$$

where  $\omega_0$  is the center (angular) frequency of the VCO and  $K_0$  is the VCO gain in  $\text{rad s}^{-1} \text{V}^{-1}$ .

[Equation \(1.1\)](#) is plotted graphically in [Fig. 1.1b](#). Because rad (radian) is a dimensionless quantity, we will drop it mostly in this text. (Note, however, that any phase variables used in this book will have to be measured in radians and not in degrees!) Therefore, in the equations a phase shift of  $180^\circ$  must always be specified as a value of  $\pi$ .

The PD (also referred to as a phase comparator) compares the phase of the output signal with the phase of the reference signal and develops an output signal  $u_d(t)$ , which is approximately proportional to the phase error  $\theta_e$ , at least within a limited range of the latter

$$u_d(t) = K_d \theta_e \quad (1.2)$$

Here,  $K_d$  represents the gain of the PD. The physical unit of  $K_d$  is  $\text{V/rad}$ . [Figure 1.1c](#) is a graphical representation of [Eq. \(1.2\)](#).

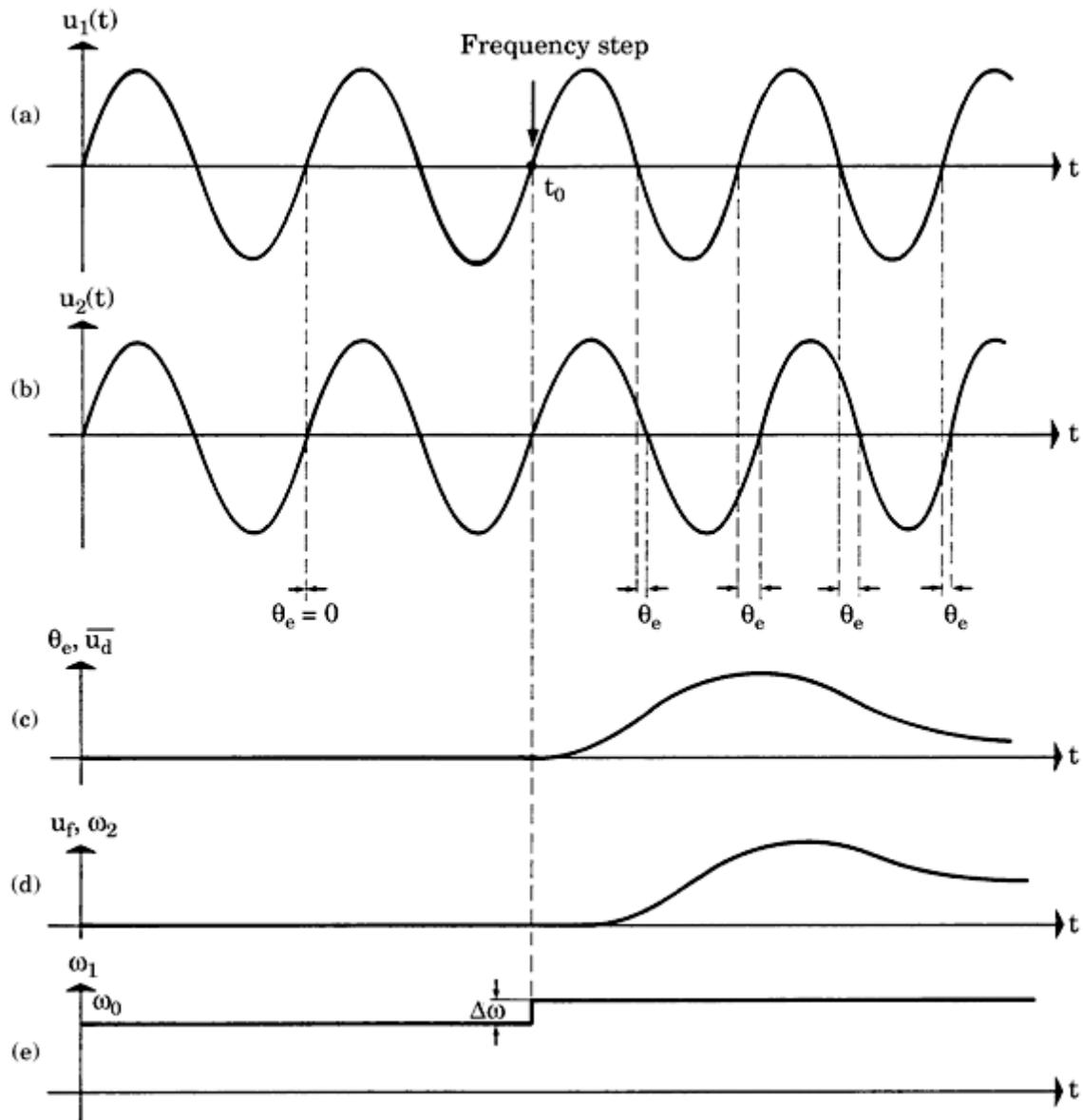
The output signal  $u_d(t)$  of the PD consists of a DC component and a superimposed AC component. The latter is undesired; hence, it is canceled by the loop filter. In most cases, a first-order low-pass filter is used. Let us now see how the three building blocks work together. First, we assume the angular frequency of the input signal  $u_1(t)$  is equal to the center frequency  $\omega_0$ . The VCO then operates at its center frequency  $\omega_0$ . As we see, the phase error  $\theta_e$  is zero. If  $\theta_e$  is zero, the output signal  $u_d$  of the PD must also be zero. Consequently, the output signal of the loop filter  $u_f$  will also be zero. This is the condition that permits the VCO to operate at its center frequency.

If the phase error  $\theta_e$  were not zero initially, the PD would develop a nonzero output signal  $u_d$ . After some delay, the loop filter would also produce a finite signal  $u_f$ . This would cause the VCO to change its operating frequency in such a way that the phase error finally vanishes.

Assume now that the frequency of the input signal is changed suddenly at time  $t_0$  by the amount  $\Delta\omega$ . As shown in [Fig. 1.2](#), the phase of the input signal then starts leading the phase of the output signal. A phase error is built up and increases with time. The PD develops a signal  $u_d(t)$ , which also increases with time. With a delay given by the loop filter,  $u_f(t)$  will also rise. This causes the VCO to increase its frequency. The phase error becomes smaller now, and after some settling time the VCO will oscillate at a frequency that is exactly the frequency of the input signal. Depending on the type of loop filter used, the final phase error will have been reduced to zero or to a finite value.

The VCO now operates at a frequency which is greater than its center frequency  $\omega_0$  by an amount  $\Delta\omega$ . This will force the signal  $u_f(t)$  to settle at a final

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.



**Figure 1.2** Transient response of a PLL onto a step variation of the reference frequency. (a) Reference signal  $u_1(t)$ . (b) Output signal  $u_2(t)$  of the VCO. (c) Signals  $\bar{u}_d(t)$  and  $\theta_e(t)$  as a function of time. (d) Angular frequency  $\omega_2$  of the VCO and loop filter output signal  $u_f(t)$  as a function of time. (e) Angular frequency  $\omega_1$  of the reference signal  $u_1(t)$ .

value of  $u_f = \Delta\omega/K_0$ . If the center frequency of the input signal is frequency-modulated by an arbitrary low-frequency signal, then the output signal of the loop filter is the demodulated signal.

The PLL can consequently be used as an FM detector. As we shall see later, it can be further applied as an AM or PM detector.

One of the most intriguing capabilities of the PLL is its ability to suppress noise

superimposed on its input signal. Let us suppose that the input signal of the PLL is buried in noise. The PD tries to measure the phase error between input and output signals. The noise at the input causes the zero crossings of the input signal  $u_1(t)$  to be advanced or delayed in a stochastic manner. This causes the PD output signal  $u_d(t)$  to jitter around an average value. If the corner frequency of the loop filter is low enough, almost no noise will be noticeable in the signal  $u_f(t)$ , and the VCO will operate in such a way that the phase of the signal

$u_2(t)$  is equal to the average phase of the input signal  $u_1(t)$ . Therefore, we can state that the PLL is able to detect a signal that is buried in noise. These simplified considerations have shown that the PLL is nothing but a servo system that controls the phase of the output signal  $u_2(t)$ .

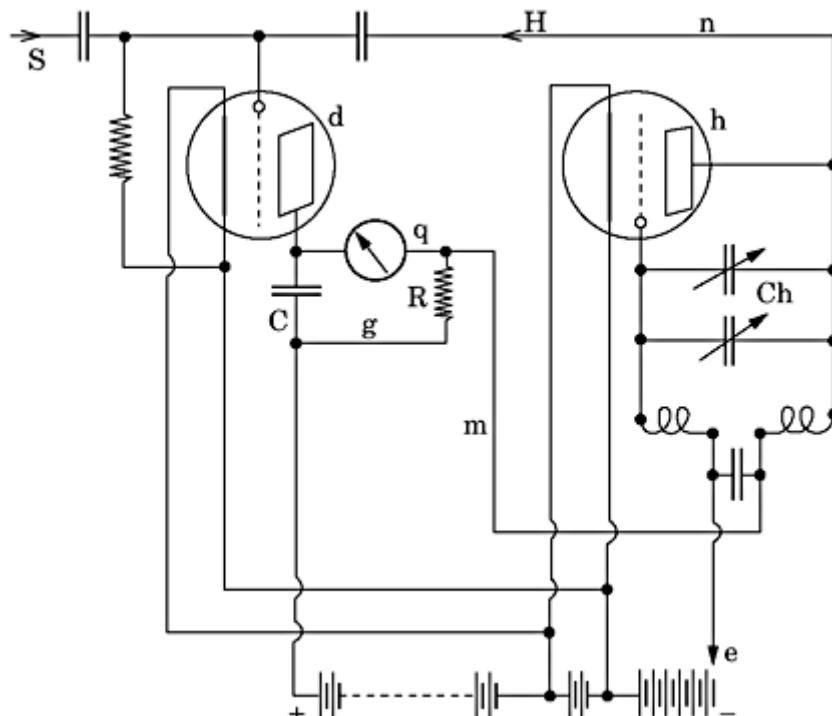
As shown in Fig. 1.2, the PLL was always able to track the phase of the output signal to the phase of the reference signal; this system was locked at all times. This is not necessarily the case, however, because a larger frequency step applied to the input signal could cause the system to “unlock.” The control mechanism inherent in the PLL will then try to become locked again, but will the system indeed lock again? We shall deal with this problem in the following chapters. Basically two kinds of problems must be considered:

- The PLL is initially locked. Under what conditions will the PLL remain locked?
- The PLL is initially unlocked. Under what conditions will the PLL become locked?

If we try to answer these questions, we notice that different PLLs behave quite differently in this regard. We find there are some fundamentally different types of PLLs. We will identify these various types in Sec. 1.3.

## Historical Background

The French engineer *Henri de Bellescize* is considered to be the inventor of the PLL. His very first implementation goes back to the year 1932. De Bellescize published his vacuum tube circuit in the French journal *L' Onde Electrique*.<sup>22</sup> The actual schematic is given in Fig. 1.3 and will probably look familiar only to a



**Figure 1.3** De Bellescize’s PLL circuit of the year 1932.

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

few veterans who started their career with building ham radios from electron tubes. The tube on the right side of the figure in combination with the LC tank circuit forms an oscillator, and as we will recognize soon, it is even a *voltage-controlled oscillator*. The output signal of the oscillator [labeled  $H$  and corresponding to  $u_2(t)$  in Fig. 1.1a] is capacitively coupled to the grid of the tube on the left. The reference signal [labeled  $S$  and corresponding to  $u_1(t)$  in Fig. 1.1a] is also fed via another capacitor to that grid. Because the grid voltage – anode current characteristic of electron tubes is nonlinear, the anode current contains a product term that is, a signal proportional to  $S \cdot H$  or  $u_1(t) \cdot u_2(t)$ . As will be shown in Sec. 2.4.1, the circuit around the left tube is a multiplier type phase detector. When the circuit is locked, this product is a measure of phase error—in other words, of the phase difference between the signals  $S$  and  $H$ . The parallel RC circuit in the anode is the *loop filter*. The voltage drop across that filter is therefore proportional to the phase error. That voltage applied to the anode of the right tube is now the difference of the battery voltage ( $e$ ) and the voltage drop across resistor  $R$ —that is, the phase error modulates the anode voltage of the oscillator. Because the frequency generated by the oscillator is an almost linear function of anode voltage, the oscillator is a VCO indeed!

This brilliant invention was widely ignored by most engineers for about 20 years. One of the first large-scale industrial applications of the PLL (back in the 1950s) was the color subcarrier recovery in color TV receivers. PLL-like circuits were also used in TV for line and frame synchronization. Somewhat later frequency synthesizers built from PLLs were used to generate a raster of frequencies in the local oscillator of FM receivers. The real breakthrough of the PLL came with desktop computers and with the PC, where PLLs are used for many types of data synchronization—for instance, reading digital data to and from floppy disks, hard disks, modems, tape drives, and the like. One of the largest applications today is probably the mobile phone, where the PLL is used again for frequency synthesis.

## Classification of PLL Types

The very first phase-locked loops (PLLs) were built from discrete components, including electron tubes and, later, discrete transistors. All these circuits were linear circuits. The first PLL ICs appeared around 1965 and were also purely analog devices. An analog multiplier (four-quadrant multiplier) was used as the phase detector, the loop filter was built from a passive or active RC filter, and the well-known voltage-controlled oscillator (VCO) was used to generate the output signal of the PLL. This type of PLL is referred to as the *linear PLL* (LPLL) today. In the years that followed, the PLL drifted slowly but steadily into digital territory. The very first digital PLL (DPLL), which appeared around 1970, was in effect a hybrid device: only the phase detector was built from a digital circuit (for instance, from an EXOR gate or a JK-flipflop), but the remaining blocks were still analog. A few years later, the “all-digital” PLL (ADPLL) was invented. The ADPLL is exclusively built from digital function blocks; hence, it doesn’t contain any passive components like resistors and capacitors.

In analogy to filters, PLLs can also be implemented “by software.” In this case, the function of the PLL is no longer performed by a piece of specialized hardware, but rather by a computer program. This last type of PLL is referred to as SPLL.

Different types of PLLs behave differently, so there is no common theory which covers all kinds of PLLs. The performance of LPLLs and DPLLs is similar, however; thus, we can develop a theory that is valid for both categories. We will deal with LPLLs and DPLLs in Chaps. 2 and 3. The term “mixed” indicates that these PLLs are mostly hybrids built from linear and digital circuits. Strictly speaking, only the DPLL is a mixed-signal circuit; the LPLL is purely analog.

The ADPLL behaves very much different from mixed-signal PLLs; hence, it is discussed in a separate chapter ([Chap. 11](#)).

The software PLL is normally implemented by a hardware platform such as a microcontroller or a digital signal processor (DSP). The PLL function is realized by software. This offers the greatest flexibility because a vast number of different algorithms can be developed. For example, an SPLL can be programmed to behave like an LPLL, a DPLL, or an ADPLL. We will deal with SPLLs in [Chap. 13](#).

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

# Mixed-Signal PLL Building Blocks

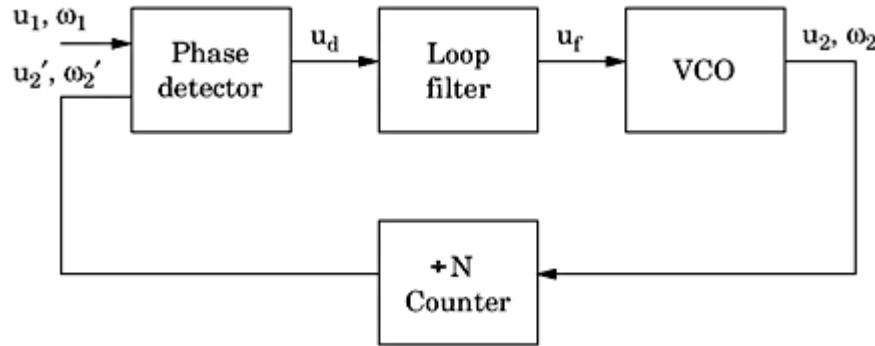
## Block Diagram of the Mixed-Signal PLL

As mentioned in [Sec. 1.3](#), the mixed-signal PLL includes circuits that are hybrids of both linear and digital circuits. To see which parts of the system are linear and which are digital, consider the general block diagram in [Fig. 2.1](#).

As shown in [Sec. 1.1](#), every PLL consists of the three blocks *phase detector*, *loop filter*, and *VCO* (voltage-controlled oscillator). When the PLL is used as a frequency synthesizer, another block is added: a *divide-by-N counter*. Assuming the counter divides by a factor  $N$ , the frequency of the VCO output signal is then forced to be  $N$  times the reference frequency (the frequency of the input signal  $u_1$ ). In most cases, the divider ratio  $N$  is made programmable. We will deal extensively with frequency synthesizers in Chaps. [6](#) and [7](#).

By inserting a down scaler, the term *center frequency* becomes ambiguous: the center (radian) frequency  $\omega_0$  can be related to the output of the VCO (as done in [Sec. 1.1](#)), but it could also be related to the output of the down scaler, or in other words, to the input of the PLL. To remove this dilemma, we introduce two different terms for center (radian) frequency: we will use the symbol  $\omega_0$  to denote the center frequency at the output of the VCO, and the symbol  $\omega_0'$  to denote the center radian frequency at the input of the PLL. Obviously,  $\omega_0$  and  $\omega_0'$  are related by  $\omega_0' = \omega_0/N$ . As shown in [Fig. 2.1](#), the quantities related to the output signal of the down scaler are characterized by a prime (' symbol)—for example,  $u_2'$ ,  $\omega_2'$ . When the VCO does not operate at its center frequency ( $u_f \neq 0$ ), its output radian frequency is denoted as  $\omega_2$ . For the down-scaled frequency, the symbol  $\omega_2'$  is used, as shown in [Fig. 2.1](#). Again, we have  $\omega_2' = \omega_2/N$ .

As will be demonstrated later in this chapter, the order (number of poles of the transfer function) of a PLL is equal to the order of the loop filter +1. In most practical PLLs, first-order loop filters are applied. These PLLs are therefore second-order systems. In a few cases, the filter may be omitted (such a PLL is a first-order loop). In this chapter, we will deal exclusively with first- and



**Figure 2.1** Block diagram of the mixed-signal PLL. The symbols defined here are used throughout this chapter. The divide-by-N counter is optional.

second-order PLLs. Higher-order loops come into play when suppression of spurious sidebands (also called “spurs”) becomes an issue. The designer of the PLL then has to provide higher-order loop filters<sup>10</sup>—that is, loop filters of order 2, 3, or even 4. Increasing the order also increases the phase shift of such filters, thus higher-order PLLs are prone to become unstable. We will deal with higher-order ( $>2$ ) PLLs in a separate chapter ([Chap. 9](#)), where we will show how to specify the poles and zeros of higher-order loop filters in order to maintain stable operation.

As was explained in [Sec. 1.1](#), the PLL is nothing more than a control system which acts on the VCO in such a way that the frequency of the signal  $u_2'$  is identical with the frequency of signal  $u_1$ . Moreover, the phase of signal  $u_2'$  is nearly identical with the phase of signal  $u_1$  or is offset by a nearly constant value from the latter. The PLL can therefore be considered as a control system for *phase signals*. Because phase signals are less frequently found in control theory than voltage or current signals, for instance, we will consider the nature of phase signals in more detail in [Sec. 2.2](#). The properties of the PLL’s building blocks will be discussed in [Sec. 2.4](#) through [2.7](#).

## A Note on Phase Signals

Dynamic analysis of control systems is normally performed by means of its transfer function  $H(s)$ .  $H(s)$  relates the input and output signals of the system; in conventional electrical networks, the input and the output are represented by voltage signals  $u_1(t)$  and  $u_2(t)$ , respectively, so  $H(s)$  is given by

$$H(s) = \frac{U_2(s)}{U_1(s)} \quad (2.1)$$

where  $U_1(s)$  and  $U_2(s)$  are the Laplace transforms of  $u_1(t)$  and  $u_2(t)$ , respectively, and  $s$  is the Laplace operator. In the case of the PLL, the input and output signals are phases, however, which is less familiar to many electronic engineers.

**Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.**

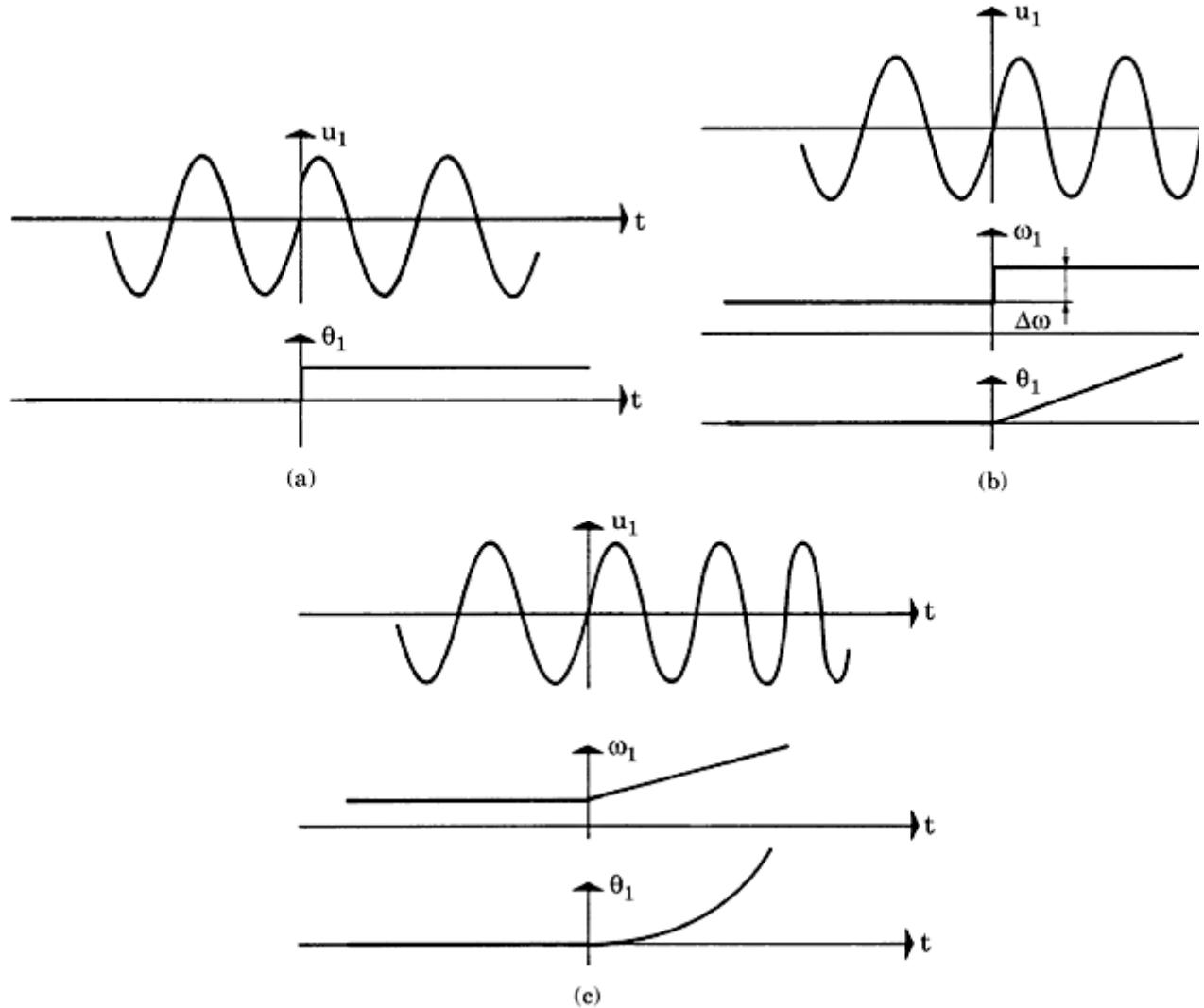
To see what phase signals really are, we assume for the moment that both input and output signals of the PLL (Fig. 2.1) are sine waves:

$$\begin{aligned} u_1(t) &= U_{10} \sin [\omega_1 t + \theta_1(t)] \\ u_2'(t) &= U_{20} \sin [\omega_2' t + \theta_2'(t)] \end{aligned} \quad (2.2)$$

The information carried by these signals is neither the amplitude ( $U_{10}$  or  $U_{20}$ , respectively) nor the frequency ( $\omega_1$  or  $\omega_2'$ , respectively) but the phases  $\theta_1(t)$  and  $\theta_2'(t)$ .

**Note:** Because we used the symbol  $\omega_2'$  for the radian frequency at the output of the down scaler (Fig. 2.1), we use the symbol  $\theta_2'$  for the phase of signal  $u_2'$  and not  $\theta_2$ ; the latter is used to specify the phase of the VCO output signal  $u_2$ .

Let us now consider some simple phase signals. Figure 2.2 lists a number of phase signals  $\theta_1(t)$ , which are frequently used to excite a PLL. Figure 2.2a shows



**Figure 2.2** Some typical exciting functions as applied to the reference input of a PLL. (a) Phase step  $\theta_1(t)$  at  $t = 0$ ;  $\theta_1(t) = \Delta\Phi \cdot u(t)$ . (b) Frequency step  $\Delta\omega$  applied at  $t = 0$ ;  $\theta_1(t) = \Delta\omega \cdot t$ . (c) Frequency starting at  $t = 0$ ;  $\theta_1(t) = \Delta\omega \cdot t^2/2$ .

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

the simplest case: the phase  $\theta_1(t)$  performs a step change at time  $t = 0$ ; hence, it is given by

$$\theta_1(t) = \Delta\Phi \cdot u(t) \quad (2.3)$$

where  $u(t)$  is the unit step function. This case is an example for phase modulation.

Let us consider next an example of frequency modulation (Fig. 2.2b). Assume the angular frequency of the reference signal is  $\omega_0'$  for  $t < 0$ . At  $t = 0$ , the angular frequency is abruptly changed by the increment  $\Delta\omega$ . For  $t \geq 0$ , the reference signal is consequently given by

$$u_1(t) = U_{10} \sin(\omega_0' t + \Delta\omega t) = U_{10} \sin(\omega_0' t + \theta_1) \quad (2.4)$$

In this case, the phase  $\theta_1(t)$  can be written as

$$\theta_1(t) = \Delta\omega \cdot t \quad (2.5)$$

Consequently, the phase  $\theta_1(t)$  is a ramp signal.

As a last example, consider a reference signal whose angular frequency is  $\omega_0'$  for  $t < 0$  and increases linearly with time for  $t \geq 0$  (Fig. 2.2c). For  $t \geq 0$ , its angular frequency is therefore

$$\omega_1(t) = \omega_0' + \Delta\dot{\omega} \cdot t \quad (2.6)$$

where  $\Delta\dot{\omega}$  denotes the rate of change of angular frequency. Remember that the angular frequency of a signal is defined as the first derivative of its phase with respect to time:

$$\omega_1(t) = \frac{d\theta_1}{dt} \quad (2.7)$$

Hence the phase of a signal at time  $t$  is the integral of its angular frequency over the time interval  $0 < \tau < t$ , where  $\tau$  denotes elapsed time. The reference signal can be written as

$$u_1(t) = U_{10} \sin \int_0^t (\omega_0' + \Delta\dot{\omega} \cdot \tau) d\tau = U_{10} \sin \left( \omega_0' t + \Delta\dot{\omega} \frac{t^2}{2} \right) \quad (2.8)$$

Consequently, the corresponding phase signal  $\theta_1(t)$  is given by

$$\theta_1(t) = \Delta\dot{\omega} \frac{t^2}{2} \quad (2.9)$$

in other words, it is a quadratic function of time.

## **Building Blocks of Mixed-Signal PLLs**

As shown in [Fig. 2.1](#), a mixed-signal PLL is set up from four different building blocks: phase detector, loop filter, controlled oscillator, and (optionally) down

scaler. We will discuss the properties of phase detectors in [Sec. 2.4](#), the loop filters in [Sec. 2.5](#), controlled oscillators in [Sec. 2.6](#), and down scalers in [Sec. 2.7](#).

## Phase Detectors

A phase detector is a circuit capable of delivering an output signal that is proportional to the phase difference between its two input signals  $u_1$  and  $u_2'$  (cf. [Fig. 2.1](#)). Many circuits could be applied. In mixed signal PLLs, mainly four types of phase detectors are used. The first phase detector in the history of the PLL was the linear *multiplier* (also referred to as four-quadrant multiplier). When the PLL moved into digital territory, digital phase detectors become popular, such as the EXOR gate, the edge-triggered *JK-flipflop* and the so-called *phase-frequency detector* (PFD). Let us start with the discussion of the multiplier phase detector.

### Type 1: Multiplier phase detectors

The multiplier phase detector is used exclusively in linear PLLs (LPLLs). In a LPLL, the input signal  $u_1$  is mostly a sine wave and is given by

$$u_1(t) = U_{10} \sin(\omega_1 t + \theta_1) \quad (2.10a)$$

where  $U_{10}$  is the amplitude of the signal,  $\omega_1$  is its radian frequency, and  $\theta_1$  its phase. The second input signal  $u_2'$  (cf. also [Fig. 2.1](#)) is usually a symmetrical square wave signal (sometimes also called *Walsh* function)<sup>21</sup> and is given by

$$u_2'(t) = U_{20} \text{rect}(\omega_2' t + \theta_2') \quad (2.10b)$$

where *rect* stands for “rectangular” (square wave), and  $U_{20}$  is the amplitude,  $\omega_2'$  the radian frequency and  $\theta_2'$  the phase. These signals are shown in [Fig. 2.3](#). The dashed curve in [Fig. 2.3a](#) is a sine wave having a phase of  $\theta_1 = 0$ ; the solid line has a nonzero phase  $\theta_1$ . For simplicity, we assume here that the phase is constant over time. The dashed curve in [Fig. 2.3b](#) shows a symmetrical square wave having a phase  $\theta_2' = 0$ ; the solid line has a nonzero phase. The output signal of the four-quadrant multiplier is obtained by multiplying the signals  $u_1$  and  $u_2'$ . To simplify the analysis, the square wave signal is replaced by its Fourier series. For  $u_2'(t)$ , we then get

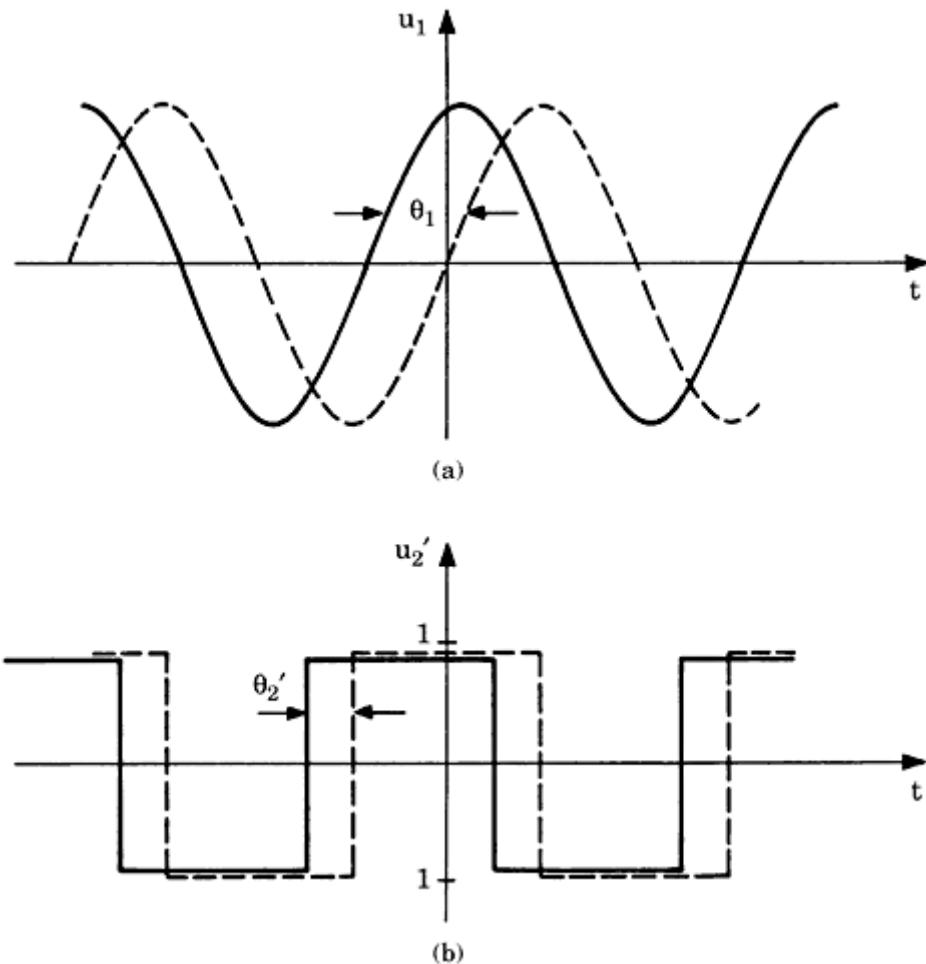
$$u_2'(t) = U_{20} \left[ \frac{4}{\pi} \cos(\omega_2' t + \theta_2') + \frac{4}{3\pi} \cos(3\omega_2' t + \theta_2') - \dots \right] \quad (2.11)$$

The first term in square brackets is the fundamental component; the remaining terms are odd harmonics. For the output signal  $u_d(t)$ , therefore we get

$$u_d(t) = u_1(t) \cdot u_2'(t) = U_{10}U_{20} \sin(\omega_1 t + \theta_1) \cdot \left[ \frac{4}{\pi} \cos(\omega_2' t + \theta_2') + \frac{4}{3\pi} \cos(3\omega_2' t + \theta_2') + \dots \right] \quad (2.12)$$

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.



**Figure 2.3** Input signals of the multiplier phase detector. (a) Signal  $u_1(t)$  is a sine wave. Dashed line: phase  $\theta_1 = 0$ ; solid line: phase  $\theta_1 > 0$ . (b) Signal  $u_2'(t)$  is a symmetrical square wave signal. Dashed line:  $\theta_2' = 0$ ; solid line:  $\theta_2' > 0$ .

When the PLL is locked, the frequencies  $\omega_1$  and  $\omega_2'$  are identical, and  $u_d(t)$  becomes

$$u_d(t) = U_{10}U_{20} \left[ \frac{2}{\pi} \sin \theta_e \right] \quad (2.13)$$

where  $\theta_e = \theta_1 - \theta_2'$  is the phase error. The first term of this series is the wanted “DC” term, whereas the higher-frequency terms will be eliminated by the loop filter. Setting  $K_d = 2U_{10}U_{20}/\pi$  and neglecting higher-frequency terms, we get

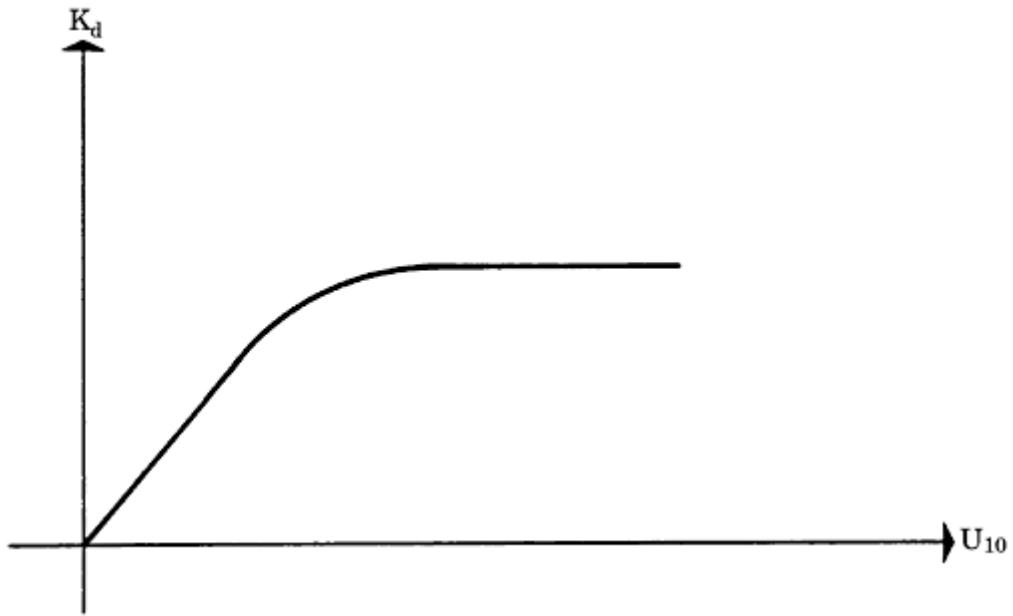
$$u_d(t) \approx K_d \sin(\theta_e) \quad (2.14)$$

where  $K_d$  is called *detector gain*. When the phase error is small, the sine function can be

replaced by its argument, and we get

$$u_d(t) \approx K_d \theta_e \quad (2.15)$$

This equation represents the linearized model of the phase detector.



**Figure 2.4** Phase-detector gain  $K_d$  as a function of the amplitude  $U_{10}$  of the reference signal.

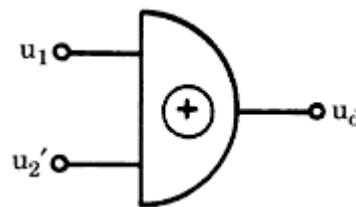
The dimension of  $K_d$  is V/rad. As shown earlier,  $K_d$  is proportional to both amplitudes  $U_{10}$  and  $U_{20}$ . Normally,  $U_{20}$  is constant, so  $K_d$  becomes a linear function of the input signal level  $U_{10}$ . This is plotted in Fig. 2.4. Because the multiplier saturates when its output signal comes close to the power supply rails, this function flattens out at large signal levels, and  $K_d$  approaches a limiting value. To conclude the analysis of the four-quadrant multiplier, we state that—in the locked state of the PLL—the phase detector represents a zero-order block having a gain of  $K_d$ .

To complete the discussion of the multiplier-type phase detector, we look at its behavior in the unlocked state of the PLL. When the PLL is out of lock, the radian frequencies  $\omega_1$  and  $\omega_2'$  are different. The output signal of the multiplier then can be written as

$$u_d(t) = K_d \sin(\omega_1 t - \omega_2' t + \theta_1 - \theta_2') + \text{higher harmonics} \quad (2.16)$$

The higher harmonics are almost entirely suppressed by the loop filter, hence there remains one AC term whose frequency equals the difference  $\omega_1 - \omega_2'$ . Because the output is an AC signal, we are tempted to conclude that its average is zero. This would imply that the average output signal of the loop filter would also be zero (cf. Fig. 2.1). This would make it impossible for the loop to acquire lock because the frequency of the VCO output signal would remain permanently hung up at its center frequency  $\omega_0$ , with a superimposed frequency modulation. As we will see in Sec. 3.9.3, however, the AC signal  $u_d(t)$  is an asymmetric “sine wave”—that is, the durations of the positive and negative half waves are different. Consequently, there will be a nonzero DC component that will pull the average output frequency of the VCO up or down until lock is acquired, provided the initial difference of radian frequencies  $\omega_1 - \omega_2'$  is

less than a limiting value called *pull-in range*  $\Delta\omega_P$ . Nevertheless, note that this



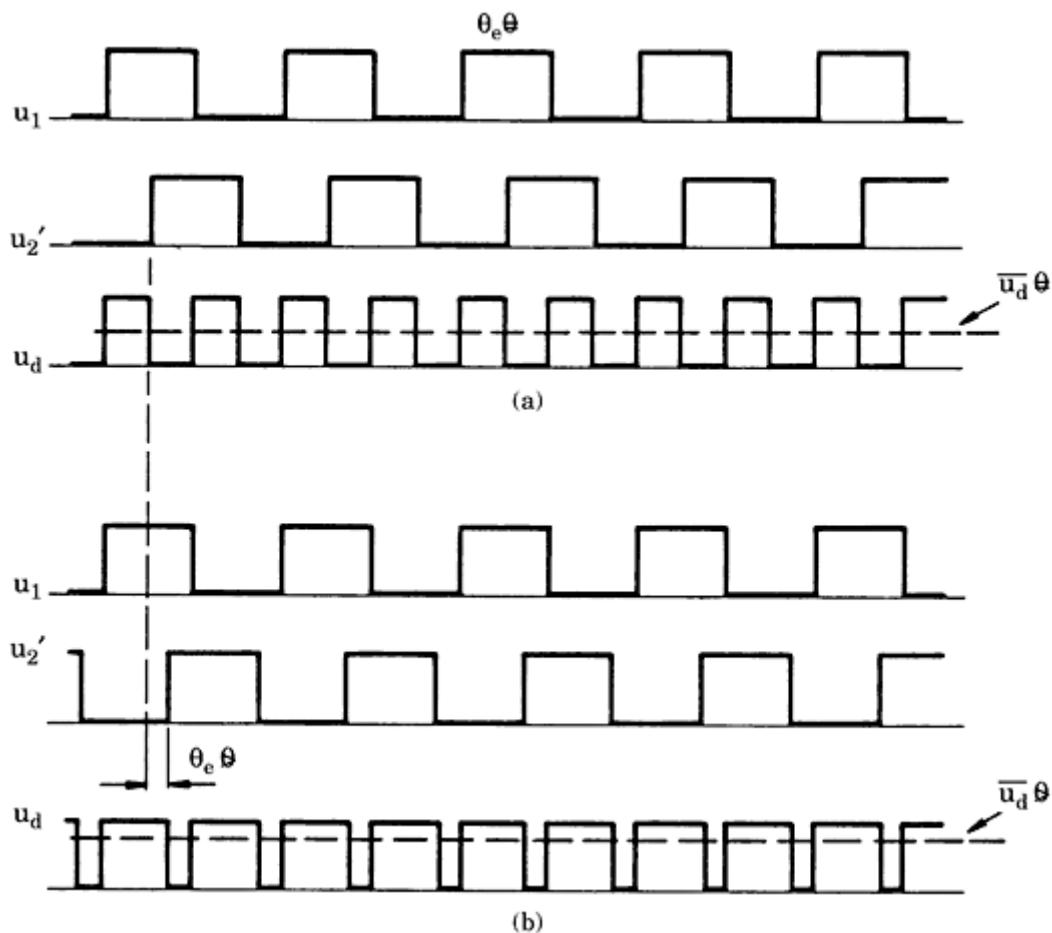
**Figure 2.5** The EXOR phase detector.

pull-in process is quite slow. It will be demonstrated later in Sec. 2.4.4 that another type of phase detector (the so-called phase-frequency detector) enables much faster acquisition, because its output signal is not only phase sensitive, but also frequency sensitive (in the unlocked state).

We continue the discussion of phase detectors with the EXOR (exclusive-OR) gate.

### Type 2: EXOR phase detectors

The operation of the EXOR phase detector (Fig. 2.5) is similar to that of the linear multiplier. The signals in DPLLS are always binary signals (for instance, square waves). We assume for the moment that both signals  $u_1$  and  $u_2'$  are symmetrical square waves. Figure 2.6 depicts the waveforms of the EXOR phase detector



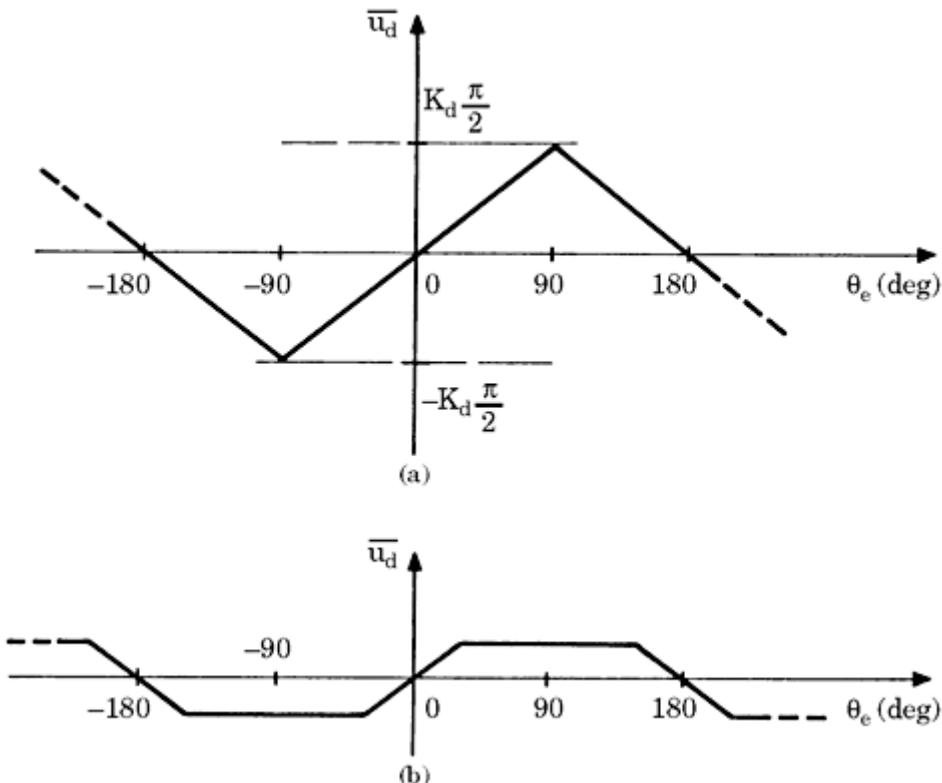
**Figure 2.6** Waveforms of the signals for the EXOR phase detector. (a) Waveforms at zero phase error ( $\theta_e = 0$ ). (b) Waveforms at positive phase error ( $\theta_e > 0$ ).

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

for different phase errors  $\theta_e$ . At zero phase error, the signals  $u_1$  and  $u_2'$  are out of phase by exactly  $90^\circ$ , as shown in Fig. 2.6a. Then the output signal  $u_d$  is a square wave whose frequency is twice the reference frequency; the duty cycle of the  $u_d$  signal is exactly 50 percent. Because the high-frequency component of this signal will be filtered out by the loop filter, we consider only the average value of  $u_d$ , as shown by the dashed line in Fig. 2.6a. The average value (denoted  $\bar{u}_d$ ) is the arithmetic mean of the two logical levels; if the EXOR is powered from an asymmetrical 5-V power supply,  $\bar{u}_d$  will be approximately 2.5 V. This voltage level is considered the quiescent point of the EXOR and will be denoted as  $\bar{u}_d = 0$  from now on. When the output signal  $u_2'$  lags the reference signal  $u_1$ , the phase error  $\theta_e$  becomes positive by definition (this case is shown in Fig. 2.6b). Now the duty cycle of  $u_d$  becomes larger than 50 percent—in other words, the average value of  $u_d$  is considered positive, as shown by the dashed line in the  $u_d$  waveform. Clearly, the mean of  $u_d$  reaches its maximum value for a phase error of  $\theta_e = 90^\circ$  and its minimum value for  $\theta_e = -90^\circ$ . If we plot the mean of  $u_d$  versus phase error  $\theta_e$ , we get the characteristic shown in Fig. 2.7a. Whereas the output signal of the four-quadrant multiplier varied with the sine of phase error, the average output of the EXOR is a triangular function of phase error. Within a phase error range of  $-90^\circ < \theta_e < 90^\circ$ ,  $\bar{u}_d$  is exactly proportional to  $\omega_e$  and can be written as

$$\bar{u}_d = K_d \theta_e \quad (2.17)$$



**Figure 2.7** Plot of averaged phase detector output signal  $\overline{u_d}$  versus phase error  $\theta_e$ . (a) Normal case: waveforms  $u_1$  and  $u_2'$  in Fig. 2.6 are symmetrical square waves. (b) Waveforms  $u_1$  and  $u_2'$  are asymmetrical. The characteristic of the phase detector is clipped.

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

In the case of the EXOR phase detector, the phase detector gain  $K_d$  is constant. When the supply voltages of the EXOR are  $U_B$  and 0, respectively, and when we assume that the logic levels are  $U_B$  and 0,  $K_d$  is given by

$$K_d = \frac{U_B}{\pi} \quad (2.18)$$

When the output signal of the EXOR does not reach the supply rails but rather saturates at some higher level  $U_{\text{sat}+}$  (in the high state) and some lower level  $U_{\text{sat}-}$  (in the low state),  $K_d$  must be calculated from

$$K_d = \frac{U_{\text{sat}+} - U_{\text{sat}-}}{\pi} \quad (2.19)$$

Like the four-quadrant multiplier, the EXOR phase detector can maintain phase tracking when the phase error is confined to the range

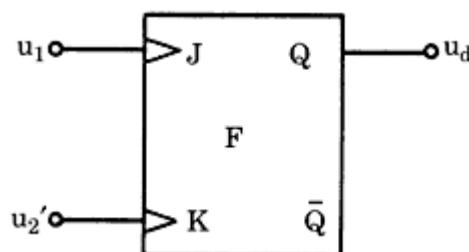
$$-\frac{\pi}{2} < \theta_e < \frac{\pi}{2}$$

The performance of the EXOR phase detector becomes severely impaired if the signals  $u_1$  and  $u_2'$  become asymmetrical. If this happens, the output signal gets clipped at some intermediate level, as shown by Fig. 2.7b. This reduces the loop gain of the PLL and results in smaller lock range, pull-out range, and so on.

It is important to also look at the performance of the EXOR phase detector in the unlocked state of the PLL, as we did in the case of the multiplier phase detector. When the PLL is out of lock, the radian frequencies  $\omega_1$  and  $\omega_2'$  are different. The output signal of the EXOR then contains an AC term whose fundamental radian frequency is the difference  $\omega_1 - \omega_2'$ , the higher harmonics of which will be filtered out by the loop filter. The EXOR therefore performs very similar to the multiplier phase detector—in other words, the pull-in process becomes slow, and acquisition is only realized when the difference  $\omega_1$  and  $\omega_2'$  is less than the pull-in frequency  $\Delta\omega_P$ . This will be discussed in more detail in Sec. 3.9.3.

### Type 3: JK-flipflop phase detectors

The JK-flipflop phase detector is shown in Fig. 2.8.



**Figure 2.8** The JK-flipflop phase detector.

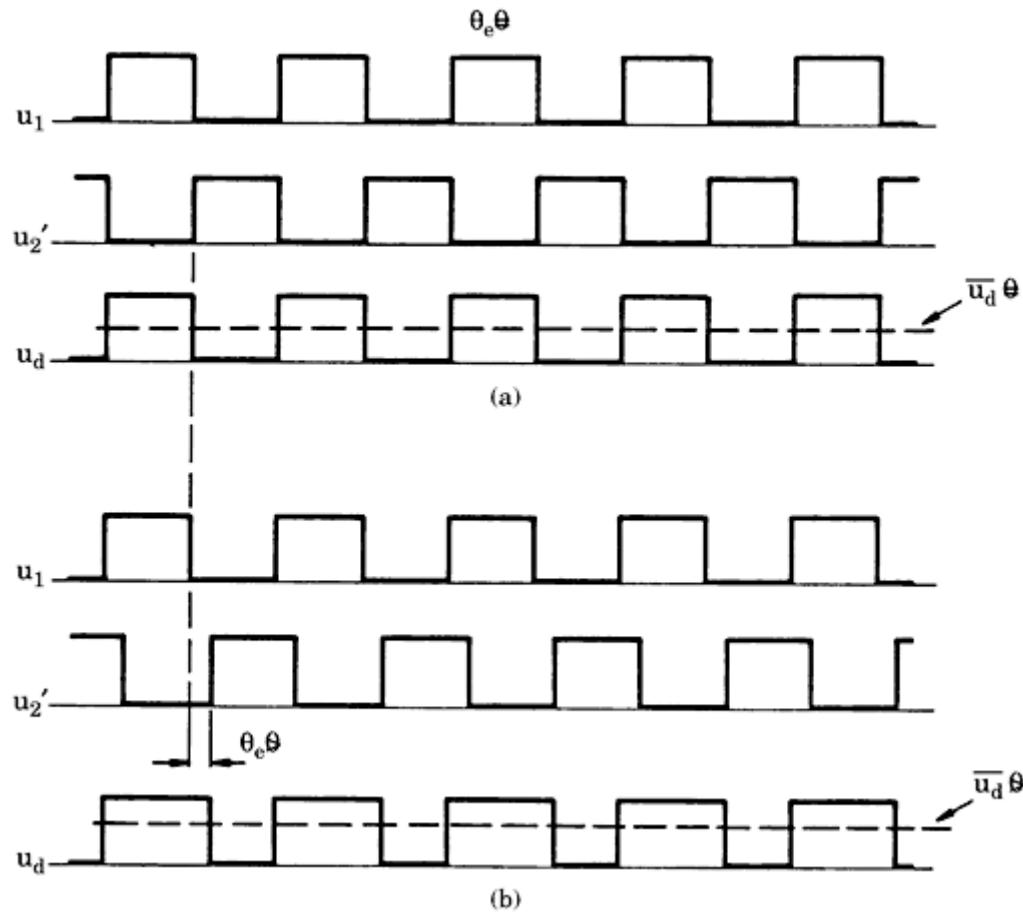
---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

This type of JK-flipflop differs from conventional JK-flipflops because it is edge-triggered. A positive edge appearing at the J input triggers the flipflop into its “high” state ( $Q = 1$ ), and a positive edge at the K input triggers the flipflop into its “low” state ( $Q = 0$ ). [Figure 2.9a](#) shows the waveforms of the JK-flipflop phase detector for the case  $\theta_e = 0$ . With no phase error,  $u_1$  and  $u_2'$  have opposite phase. The output signal  $u_d$  then represents a symmetrical square wave whose frequency is identical with the reference frequency (and not twice the reference frequency). This condition is considered as  $\bar{u}_d$  being zero. If the phase error becomes positive ([Fig. 2.9b](#)) the duty cycle of the  $u_d$  signal becomes greater than 50 percent—in other words,  $\bar{u}_d$  becomes positive. Clearly,  $\bar{u}_d$  becomes maximum when the phase error reaches  $180^\circ$  and minimum when the phase error is  $-180^\circ$ . If the mean value of  $u_d$  is plotted versus phase error  $\theta_e$ , the sawtooth characteristic of [Fig. 2.10](#) is obtained.

Within a phase error range of  $-\pi < \theta_e < \pi$ , the average signal  $\bar{u}_d$  is proportional to  $\theta_e$  and is given by

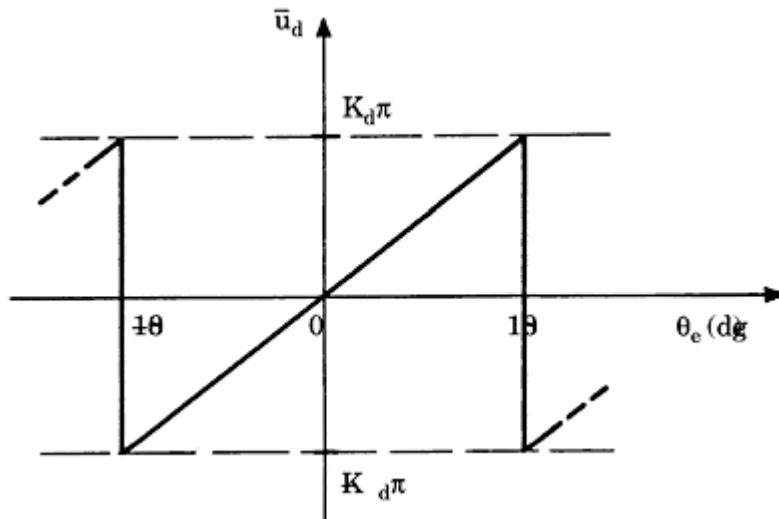
$$\bar{u}_d = K_d \theta_e \quad (2.20)$$



**Figure 2.9** Waveforms of the signals for the JK-flipflop phase detector. (a) Waveforms at zero phase error. (b) Waveforms at positive phase error.

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.



**Figure 2.10** Plot of averaged phase detector output signal  $\bar{u}_d$  versus phase error  $\theta_e$ . In contrast to the EXOR,  $\bar{u}_d$  does not depend on the duty cycle of the signals.

Obviously, the JK-flipflop phase detector is able to maintain phase tracking for phase errors within the range  $-\pi < \theta_e < \pi$ . By an analogous consideration, the phase detector gain of the JK-flipflop phase detector is given by

$$K_d = \frac{U_B}{2\pi} \quad (2.21)$$

when the logic levels are  $U_B$  or 0, respectively. If, however, these levels are limited by saturation, phase detector gain must be computed from

$$K_d = \frac{U_{\text{sat+}} - U_{\text{sat-}}}{2\pi} \quad (2.22)$$

In contrast to the EXOR gate, the symmetry of the  $u_1$  and  $u_2'$  signals is irrelevant, because the state of the JK-flipflop is altered only by the positive transitions of these signals.

In the unlocked state, the JK-flipflop phase detector behaves very much the same as the EXOR and the multiplier phase detectors. When radian frequencies  $\omega_1$  and  $\omega_2'$  are different, the output signal  $u_d(t)$  of the JK-flipflop phase detector contains a term whose fundamental radian frequency is  $\omega_1 - \omega_2'$ . Higher harmonics are removed by the loop filter. The pull-in process of the PLL containing a JK-flipflop phase detector will be discussed in more detail in Sec. 3.9.3.

#### Type 4: Phase-frequency detectors (PFDs)

The PFD is the most widely used type of phase detector. As the name implies, its output signal does not only depend on a phase error but also on a frequency

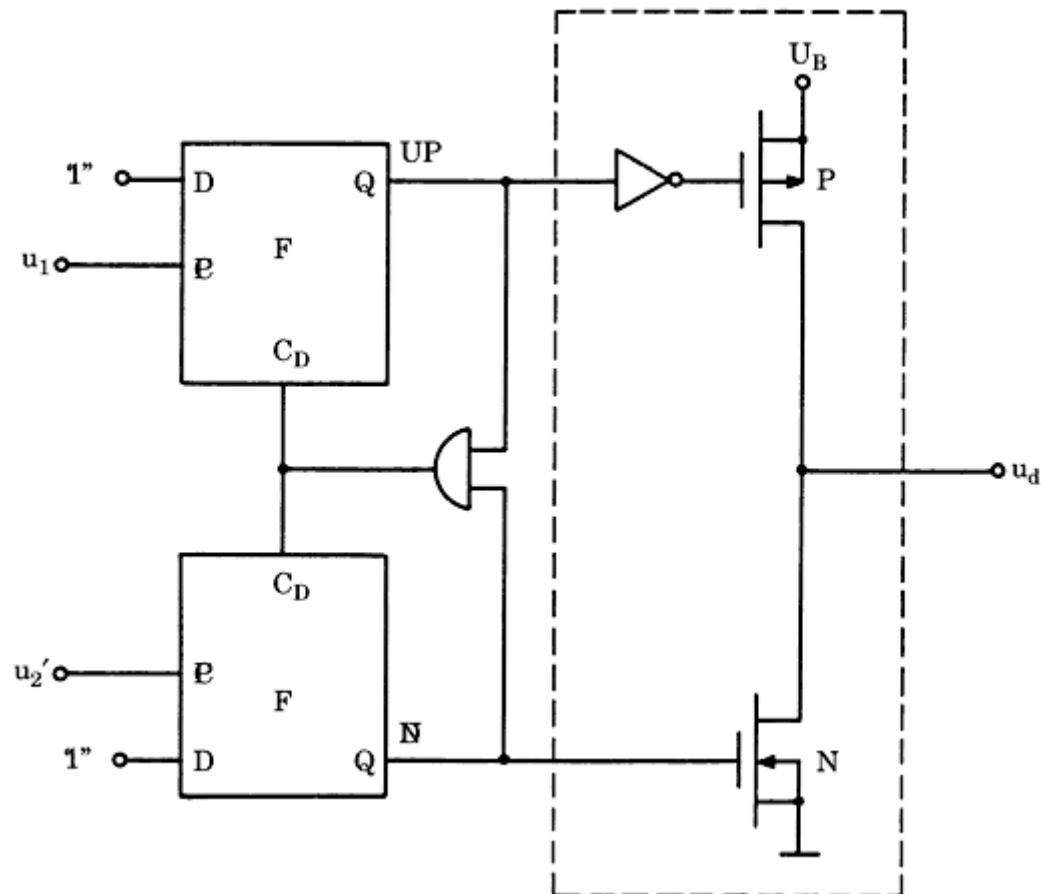
---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

error. Because the PFD “knows” at all times whether the frequency of input signal  $u_1(t)$  is higher or lower than the frequency of (scaled-down) output signal  $u_2'(t)$ , it allows the PLL to get locked in the most adverse situations—that is, for arbitrarily large frequency offsets between the two input signals. Two different types of PFDs are in use today: PFDs with voltage output and PFDs with current output. The latter is often referred to as “charge pump.” Besides its large benefits, the PFD with voltage output has a serious problem called “backlash.” The backlash phenomenon leads to undesired spurious frequencies (commonly referred to as “spurs”) in the output signal of frequency synthesizers (this will be discussed in more detail in [Sec. 6.7.3](#)). The backlash problem can be avoided when PFDs with current output are used. This is the reason why the PFD with current output is preferred today in most applications. We start the discussion of PFDs with the voltage output PFD.

**Type 4a: PFDs with voltage output.** The schematic diagram of the PFD is shown in [Fig. 2.11](#).

The PFD differs greatly from the phase detector types discussed hitherto. As its name implies, its output signal depends not only on phase error  $\theta_e$ , but also on frequency error  $\Delta\omega = \omega_1 - \omega_2'$ , when the PLL has not yet acquired lock. The



**Figure 2.11** Schematic diagram of the PFD with voltage output.

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

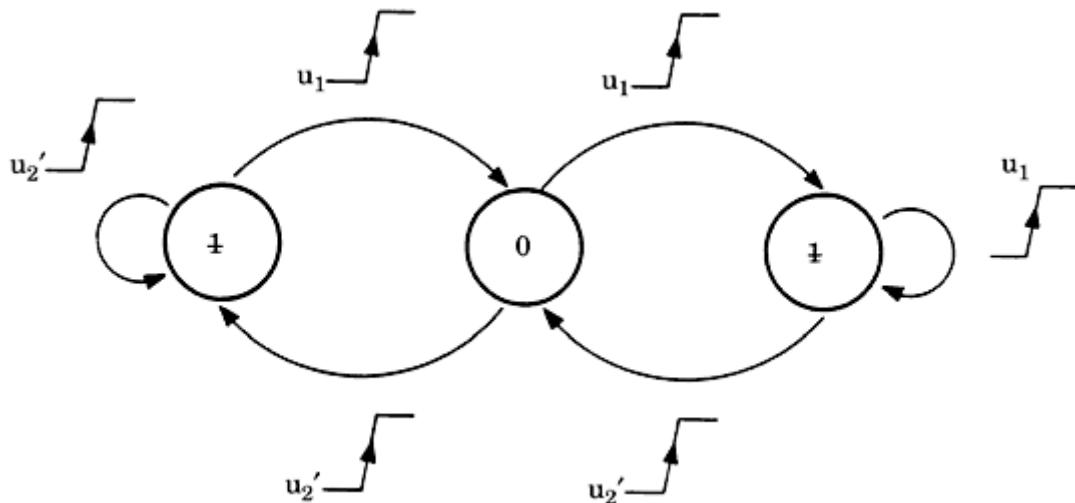
PFD is built from two D-flipflops, whose outputs are denoted “UP” and “DN” (down), respectively. The PFD can be in one of four states:

- UP = 0, DN = 0
- UP = 1, DN = 0
- UP = 0, DN = 1
- UP = 1, DN = 1

The fourth state is inhibited, however, by an additional AND gate. Whenever both flipflops are in the 1 state, a logic “high” level appears at their CD (“clear direct”) inputs, which resets both flipflops. Consequently, the device acts as a tristable device (“triflop”). We assign the symbols  $-1$ ,  $0$ , and  $1$  to these three states:

- DN = 1, UP  $\rightarrow$  0 : state =  $-1$
- UP = 0, DN  $\rightarrow$  0 : state =  $0$
- UP = 1, DN  $\rightarrow$  0 : state =  $1$

The actual state of the PFD is determined by the positive-going transients of the signals  $u_1$  and  $u_2'$ , as explained by the state transition diagram of Fig. 2.12. (In this example, we assumed the PFD acts on the positive edges of these signals exclusively; we could have reversed the definition by saying the PFD acts on the negative transitions only.) As Fig. 2.12 shows, a positive transition of  $u_1$  forces the PFD to go into its next higher state, unless it is already in the 1 state. In analogy, a positive edge of  $u_2'$  forces the PFD into its next lower state, unless it is already in the  $-1$  state. The output signal  $u_d$  is a logical function of the PFD state. When the PFD is in the 1 state,  $u_d$  must be positive; when it is in the  $-1$  state,



**Figure 2.12** A state diagram for the phase-frequency detector (PFD). This drawing shows the events causing the PFD to change its current state.

$u_d$  must be negative; and when it is in the 0 state,  $u_d$  must be zero. Theoretically,  $u_d$  is a ternary signal. Most logical circuits used today generate binary signals, but the third state ( $u_d = 0$ ) can be substituted by a “high-impedance” state.

The circuitry within the dashed box of Fig. 2.11 shows how the  $u_d$  signal is generated. When the UP signal is high, the P-channel MOS transistor conducts, so  $u_d$  equals the positive supply voltage  $U_B$ . When the DN signal is high, the N-channel MOS transistor conducts, so  $u_d$  is on ground potential. If neither signal is high, both MOS transistors are off, and the output signal floats—in other words, is in the high-impedance state. Consequently, the output signal  $u_d$  represents a tristate signal.

To see how the PFD works in a real PLL system, we consider the waveforms in Fig. 2.13. Figure 2.13a shows the (rather theoretical) case where the phase error is zero. It is assumed the PFD has been in the 0 state initially. The signals  $u_1$  and  $u_2'$  are “exactly” in phase here; both positive edges of  $u_1$  and  $u_2'$  occur “at the same time”; hence, their effects will cancel. The PFD then will stay in the 0 state forever.

Figure 2.13b shows the case where  $u_1$  leads. The PFD now toggles between the states 0 and 1. If  $u_1$  lags (as shown in Fig. 2.13c), the PFD toggles between states -1 and 0. It is easily seen from the waveforms in Fig. 2.13b and c that  $u_d$  becomes largest when the phase error is positive and approaches  $360^\circ$  (Fig. 2.13b), and smallest when the phase error is negative and approaches  $-360^\circ$  (Fig. 2.13c). If we plot the average signal  $\overline{u_d}$  versus phase error  $\theta_e$ , we get a sawtooth function, as shown in Fig. 2.14. Figure 2.14 also shows the average detector output signal for phase errors greater than  $2\pi$  or smaller than  $-2\pi$ . When the phase error  $\theta_e$  exceeds  $2\pi$ , the PFD behaves as if the phase error recycled at zero; hence, the characteristic curve of the PFD becomes periodic with period  $2\pi$ . An analogous consideration can be made for phase errors smaller than  $-2\pi$ . When the phase error is restricted to the range  $-2\pi < \theta_e < 2\pi$ , the average signal  $\overline{u_d}$  becomes

$$\overline{u_d} = K_d \theta_e \quad (2.23)$$

In analogy to the JK-flipflop, phase detector gain is computed by

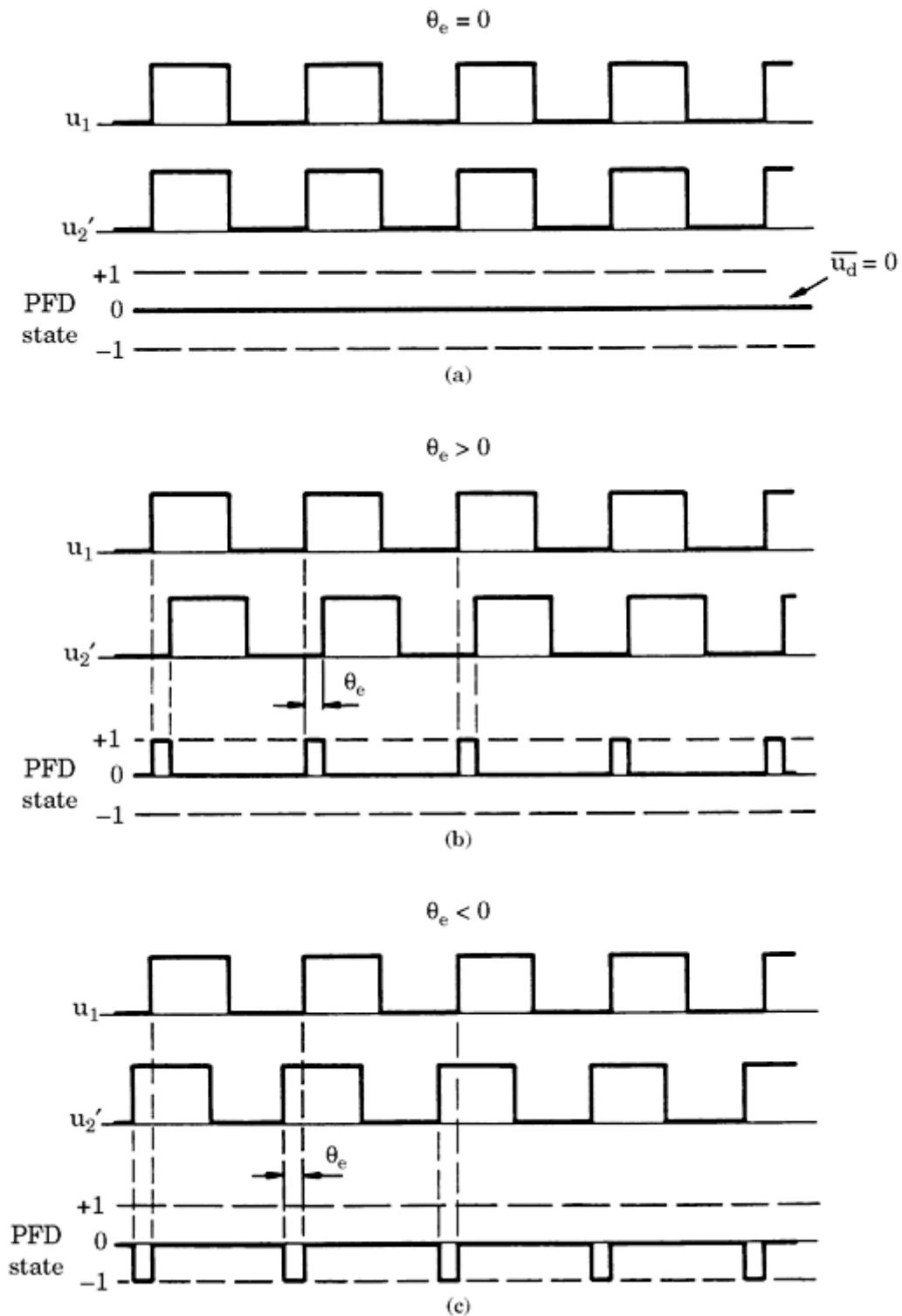
$$K_d = \frac{U_B}{4\pi} \quad (2.24)$$

when the logic levels are  $U_B$  or 0, respectively. If, however, these levels are limited by saturation, phase detector gain must be computed from

$$K_d \frac{U_{\text{sat+}} - U_{\text{sat-}}}{4\pi} \quad (2.25)$$

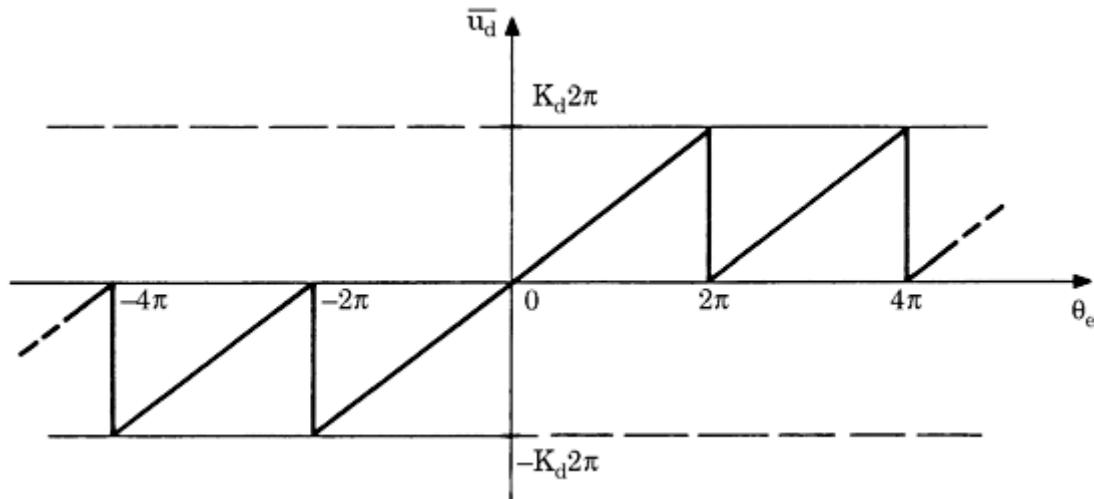
A comparison of the PFD characteristic (Fig. 2.14) with the characteristic of the JK-flipflop (Fig. 2.10) does not yet reveal exciting properties. To recognize the bonus offered by the PFD, we must assume the PLL is unlocked initially. Furthermore, we make the assumption that the

reference frequency  $\omega_1$  is higher



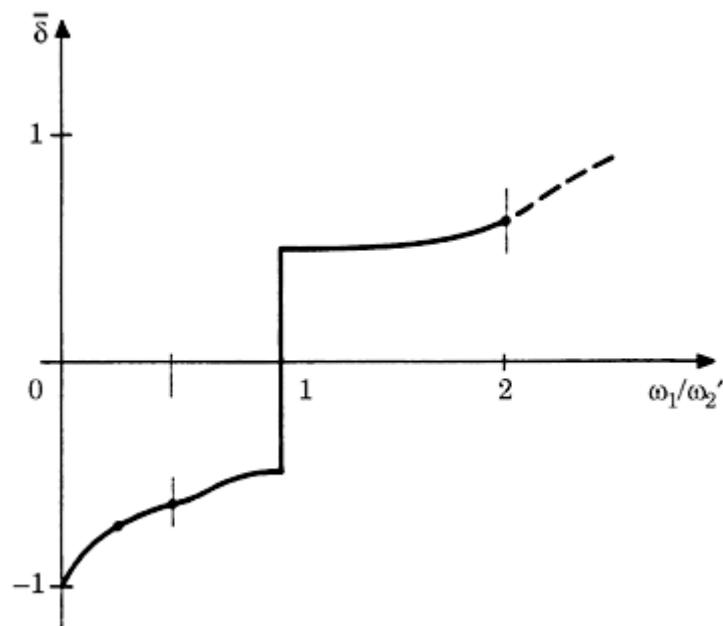
**Figure 2.13** Waveforms of the signals for the PFD. (a) Waveforms for zero phase error. The output signal of the PFD is permanently in the 0 state (high impedance). (b) Waveforms for positive phase error. The PFD output signal is pulsed to the 1 state. (c) Waveforms for negative phase error. The PFD output signal is pulsed to the -1 state.

than the output frequency  $\omega_2'$ . The  $u_1$  signal then generates more positive transitions per unit of time than the signal  $u_2'$ . Looking at Fig. 2.12, we see that the PFD can toggle only between the states 0 and 1 under this condition but will never go into the -1 state. If  $\omega_1$  is much higher than  $\omega_2'$  furthermore, the PFD



**Figure 2.14** Plot of the averaged PFD output signal  $\bar{u}_d$  versus phase error  $\theta_e$ .  $\bar{u}_d$  does not depend on the duty cycle of  $u_1$  and  $u_2'$ .

will be in the 1 state most of the time. When  $\omega_1$  is smaller than  $\omega_2'$ , however, the PFD will toggle between the states -1 and 0. When  $\omega_1$  is much lower than  $\omega_2'$ , the PFD will be in the -1 state most of the time. We conclude, therefore, that the average output signal  $u_d$  of the PFD varies monotonically with the frequency error  $\Delta\omega = \omega_1 - \omega_2'$  when the PLL is out of lock. This leads to the term *phase-frequency detector*. It is possible to calculate the duty cycle of the  $u_d$  signal as a function of the frequency ratio  $\omega_1/\omega_2'$ .<sup>23</sup> The result of this analysis is shown in Fig. 2.15. For the case  $\omega_1 > \omega_2'$ , the duty cycle  $\delta$  is defined as the average fraction



**Figure 2.15** Plot of the averaged duty cycle of the PFD output signal  $\overline{u_d}$  versus frequency ratio  $\omega_1/\omega_2'$ . This curve depicts the behavior of the PFD in the unlocked state of the DPLL.

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

of time the PFD is in the 1 state; for  $\omega_1 < \omega_2'$ ,  $\delta$  is by definition minus the average fraction of time the PFD is in the -1 state. As expected,  $\delta$  approaches 1 when  $\omega_1 \gg \omega_2'$ , and -1 when  $\omega_1 \ll \omega_2'$ . Furthermore,  $\delta$  is nearly 0.5 when  $\omega_1$  is greater than  $\omega_2'$  but both frequencies are close together, and  $\delta$  is nearly -0.5 when  $\omega_1$  is lower than  $\omega_2'$  but both frequencies are close together. This property will greatly simplify the determination of the pull-in range ([Sec. 3.9.3](#)).

It must be emphasized that no such characteristic (see [Fig. 2.15](#)) can be defined for the EXOR and for the JK-flipflop. Because the output signal  $u_d$  of the PFD depends on phase error in the locked state of the PLL and on the frequency error in the unlocked state, a PLL which uses the PFD will lock under any condition, irrespective of the type of loop filter used. For this reason, the PFD is the preferred phase detector in PLLs.

**Type 4b: PFDs with concurrent output (charge pump).** Another benefit of the PFD with voltage output is worth mentioning. In the vast majority of applications, passive lead-lag loop filters are used when the PFD serves as phase detector. When none of the MOS transistors is conducting (refer to [Fig. 2.11](#)), the output of the PFD is in the high impedance (hi-Z) state. When the PLL is locked, this is the case most of the time. Assume now that the loop filter in [Fig. 2.17a](#) is connected with the PFD output. During intervals where the PFD is in the hi-Z state, capacitor  $C_1$  cannot discharge. Hence, the passive filter behaves like a true integrator when driven from a three-state source. It is therefore unnecessary to use the more complex active filters, as described in [Sec. 2.5](#).

The schematic of the PFD with current output is shown in [Fig. 2.16](#).

The logic circuitry is the same as for the PFD with voltage output (cf. [Fig. 2.11](#)), but the complementary inverter circuit consisting of an  $N$ - and a  $P$ -channel MOSFET is replaced by two current sources. The upper one in [Fig. 2.16](#) is sourcing current  $i_d$  to the output, while the lower one is sinking current  $i_d$  from that output. The upper current source is active whenever the UP flipflop is in the 1 state, and the lower is active whenever the DN flipflop is in the 1 state. If we denote the amplitude of the current sources  $I_P$ , the average current flowing into the load is given by

$$i_d = I_P \frac{\theta_e}{2\pi} \quad (2.26)$$

When the UP flipflop is permanently ON, current  $I_P$  is sourced into the load ( $\theta_e = 2\pi$ ); when the DN flipflop is permanently ON, current  $I_P$  is sunk from the load ( $\theta_e = -2\pi$ ). In analogy to the PFD with voltage output, we define the detector gain of the current output PFD as

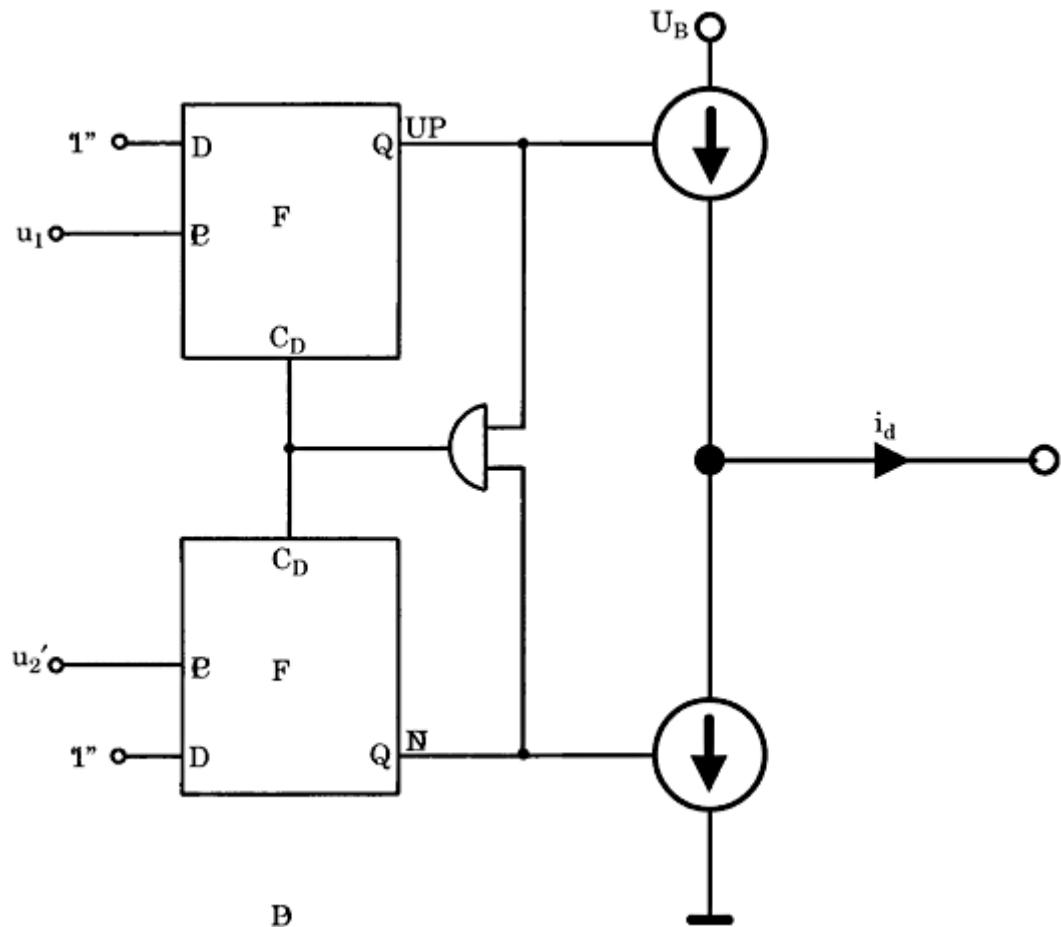
$$K_P = \frac{I_P}{2\pi} \text{ (A/rad)} \quad (2.27)$$

hence we have

$$i_d = K_P \theta_e \quad (2.28)$$

---

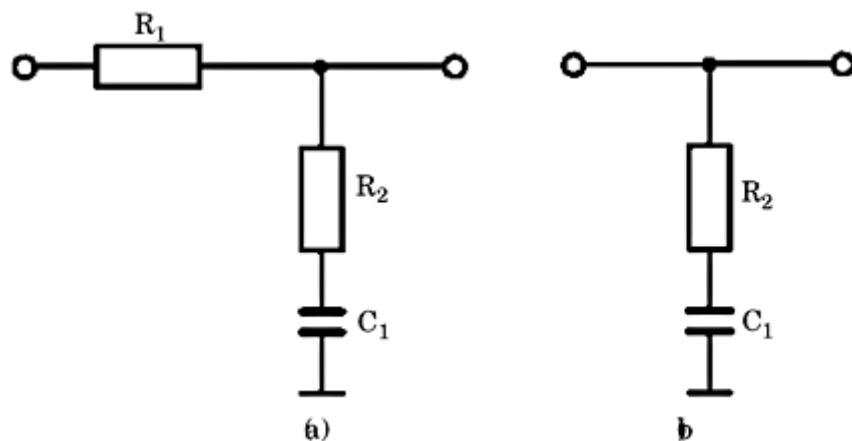
Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.



**Figure 2.16** Block diagram of the PFD with current output (charge pump).

The dimension of  $K_P$  is A/rad.  $I_P$  is typically in the range 0.1 to 2 mA in commercially available PLLs, hence  $K_P$  is typically in the range 0.01 to 0.5 mA/rad.

The question may arise now why the current output PFD should offer superior performance to the voltage output PFD. Assume that signal  $u_1(t)$  is leading  $u_2'(t)$  and that both flipflops in Fig. 2.16 are initially reset. The next positive



**Figure 2.17** First-order passive lead-lag filters. (a) For phase detectors with voltage output.  
(b) For phase detectors with current output.

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

transition of  $u_1$  then sets the UP flipflop. When  $u_2'$  goes high thereafter, the DN flipflop will also be set, thus both flipflops are momentarily in the 1 state. This forces the output of the AND gate into the 1 state, and both flipflops will be reset. Now suppose there is a delay  $t_d$  between the output of the AND gate and the clear ( $C_D$ ) inputs of the flipflops. If this delay is chosen 15 ns, for example, both flipflops will remain ON simultaneously for 15 ns upon the positive transition of  $u_2'$ . Because both current sources carry the same current, the net current flowing into the load during this interval is zero. But what is the reason to have both flipflops set at the same time? To answer this question, assume that the phase error is extremely small so the time delay between the positive transitions of  $u_1$  and  $u_2'$  is very short—for example, only 5 ps. Let us call that time delay  $t_e$ . When the positive transition in  $u_1$  occurs at time  $t_0$ , the UP flipflop is set to the 1 state at  $t_0$ . The DN flipflop is set then to the 1 state at  $t_0 + t_e$ . Because the reset signal is delayed by  $t_d$ , both flipflops are reset at  $t_0 + t_e + t_d$ . Consequently, the UP flipflop is set during a time interval  $t_e + t_d$ , and the DN flipflop is set during a time interval  $t_d$ . The net charge flowing into the load depends on  $t_e$  alone, of course, because both currents flowing when both flipflops are set cancel. For the values chosen earlier ( $t_e = 5$  ps,  $t_d = 15$  ns), the UP flipflop will be ON for 15.005 ns, but the DN flipflop is ON during only 15.000 ns. Hence, the net charge flowing into the load is the peak current multiplied by a mere 5 ps. We recognize that the current output PFD is able to react properly onto even the smallest phase error—in other words, to inject arbitrarily small amounts of charge into the load.

Let us see now how the voltage output PFD (cf. Fig. 2.11) performs under the same conditions. The positive transition of  $u_1$  first sets the UP flipflop. The output signal of the flipflop will not instantaneously swing to the positive supply voltage  $U_B$  but will start rising after some propagation delay  $t_P$  and approach the final level with a finite slope given by rise time  $t_R$ . After  $t_e$  seconds, the positive transition of  $u_2'$  occurs, which causes the output signal of the DN flipflop to go positive as well after the propagation delay  $t_P$ . When both flipflop outputs have reached a level that is sufficient to switch the AND gate to the 1 state, both flipflops start to get reset, and the output signal of both flipflops returns toward ground. When the delay  $t_e$  is shorter than the sum of propagation delay  $t_P$  and rise time  $t_R$  neither the  $P$ -channel nor the  $N$ -channel MOS transistor can turn ON, and the output of the PFD remains in the high impedance state all the time. In other words, the PFD is unable to react to small phase errors. Only when the delay  $t_e$  becomes larger than the sum of propagation delay and rise time of the D flipflop (followed by a MOS transistor) will the PFD output become active. For standard CMOS circuits, the minimum  $t_e$  value is in the order of some nanoseconds. This phenomenon is called *backlash*, and it leads to undesired sidebands in the spectrum of the VCO output signal. These sidebands are commonly referred to as *spurious frequencies*, *spurs*, or *tones*. We will deal with these spurs in greater detail in Sec. 6.7.3.

## Loop Filters (First Order)

As we have seen in Sec. 2.4, the output signal  $u_d(t)$  of the phase detector consists of a number

of terms. In the locked state of the PLL, the first of these is a

DC component that is roughly proportional to the phase error  $\theta_e$ ; the remaining terms are AC components having frequencies of  $2\omega_1$ ,  $4\omega_1$ ...

Because these higher frequencies are unwanted signals, they are filtered out by the loop filter. Because the loop filter must pass the lower frequencies and suppress the higher, it must be a low-pass filter. In most PLL designs, a first-order low-pass filter is used. Three different types of loop filters are mostly used in PLL circuits: the passive lead-lag filter, the active lead-lag filter, and the active PI filter. These will be discussed in Secs. 2.5.1 through 2.5.3.

Every loop filter is driven by a phase detector. We have seen there are phase detectors with voltage output and those with current output. The loop filters used in combination with a phase detector having a voltage output differs slightly from a loop filter designed to be used with a charge pump phase detector. Hence, all the filters discussed in the following sections will be shown with two versions.

### Type 1: Passive lead-lag filters

[Figure 2.17](#) shows first-order passive lead-lag filters having one pole and one zero. [Figure 2.17a](#) is the version used in combination with a voltage output phase detector. Its transfer function  $F(s)$  is given by

$$F(s) = \frac{\overline{U_f(s)}}{\overline{U_d(s)}} = \frac{1 + s\tau_2}{1 + s(\tau_1 + \tau_2)} \quad (2.29a)$$

where  $\tau_1 = R_1 C$  and  $\tau_2 = R_2 C$ .  $F(s)$  is the ratio of the Laplace transform of the averaged filter output signal  $\overline{u_f(t)}$  and of the Laplace transform of the averaged phase detector output voltage signal  $\overline{u_d(t)}$ . (A note on terminology: a lead-lag [also called lag-lead] filter combines a phase-leading with a phase-lagging network. The phase-leading action comes from the numerator [in other words, from the zero] in the transfer function in [Eq.\(2.29a\)](#), whereas the denominator [that is, the pole] produces the phase lag. All filters used as loop filters are lead-lag filters).

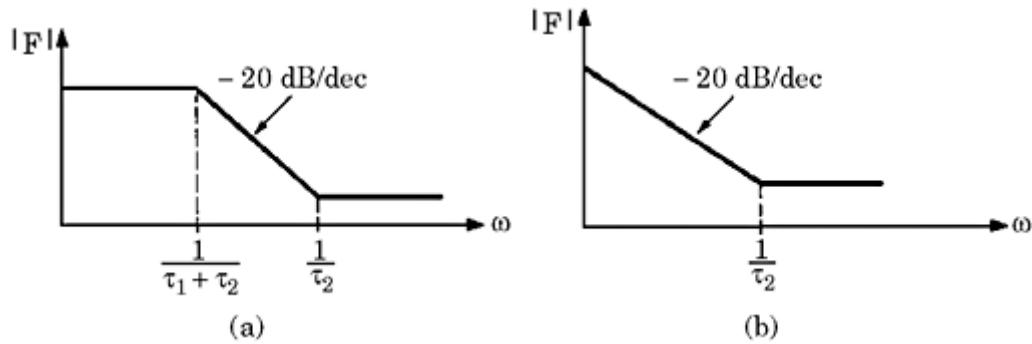
When a phase detector with current output drives the loop filter resistor,  $R_1$  becomes useless, so it is sufficient to use a loop filter as shown in [Fig. 2.17b](#). For this loop filter, the transfer function is

$$F(s) = \frac{\overline{U_f(s)}}{\overline{I_d(s)}} = \frac{1 + s\tau_2}{sC_1} \quad (2.29b)$$

Here,  $F(s)$  is the ratio of the Laplace transforms of output voltage  $u_f$  and input current  $i_d$ ; thus, it has the dimension *Ohm*.

The amplitude response of the passive lead-lag loop filters is shown in [Fig. 2.18](#). [Figure 2.18a](#) is the Bode diagram of the voltage-driven filter, while [Fig. 2.18b](#) is the Bode diagram of the current-driven filter. As we will see in [Sec. 3.4](#), the zero of this filter is crucial because it has a strong influence on the damping factor  $\zeta$  of the PLL system.

**Any use is subject to the Terms of Use as given at the website.**



**Figure 2.18** The amplitude response of first-order passive lead-lag filters. (a) A voltage-driven loop filter. (b) A current-driven loop filter.

### Type 2: Active lead-lag filters

[Figure 2.19](#) shows the first-order active lead-lag filters. In [Fig. 2.19a](#), the voltage-driven version is shown, while in [Fig. 2.19b](#), the current-driven version is displayed.

The transfer function of the voltage-driven filter is given by

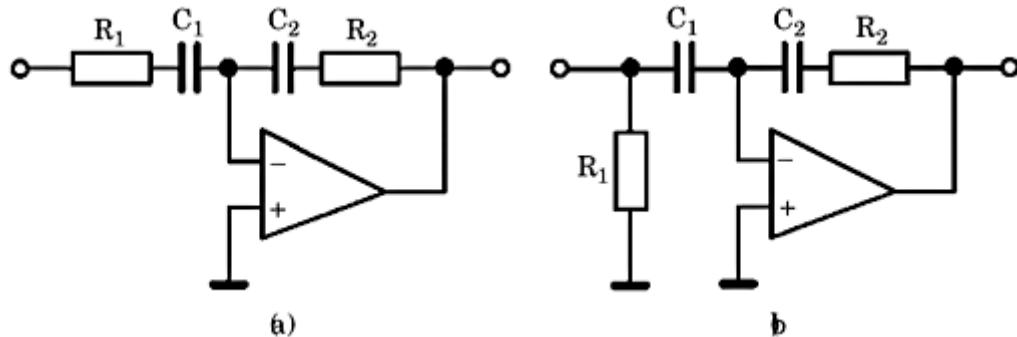
$$F(s) = K_a \frac{1 + s\tau_2}{1 + s\tau_1} \quad (2.30a)$$

where  $\tau_1 = R_1 C_1$ ,  $\tau_2 = R_2 C_2$ , and  $K_a = C_1/C_2$ . Note that this filter has a DC gain  $K_a$ , which can be made larger than 1. As will be shown in [Sec. 3.4](#), a larger gain leads to larger loop bandwidth. For the current-driven active lead-lag filter, the transfer function becomes

$$F(s) = R_1 K_a \frac{1 + s\tau_2}{1 + s\tau_1} \quad (2.30b)$$

with  $\tau_1 = R_1 C_1$ ,  $\tau_2 = R_2 C_2$ , and  $K_a = C_1/C_2$ . Note the inversion of polarity has been discarded in Eqs. [\(2.30a\)](#) and [\(2.30b\)](#).

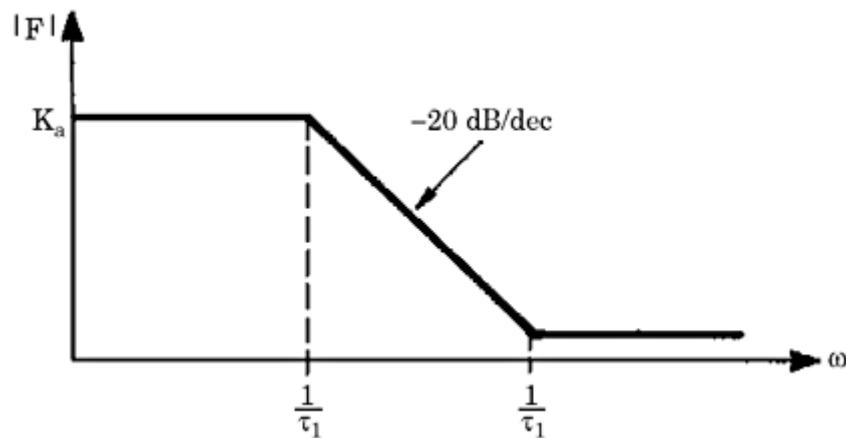
The amplitude response of the voltage-driven active lead-lag filter is given in [Fig. 2.20](#).



**Figure 2.19** First-order active lead-lag filters. (a) The version for voltage-driven input. (b) The version for current-driven input.

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.



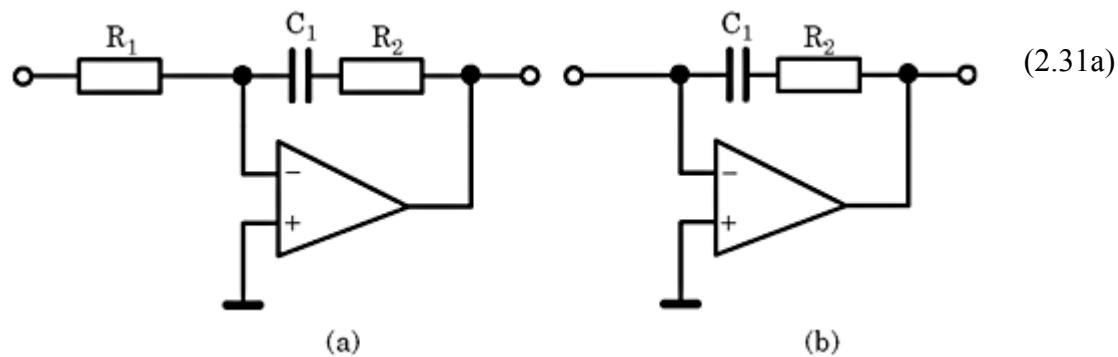
**Figure 2.20** The amplitude response of the voltage-driven first-order active lead-lag filter.

The amplitude response of the current-driven version looks very much the same, with the exception that the DC gain is  $R_1 K_a$  now and not  $K_a$ .

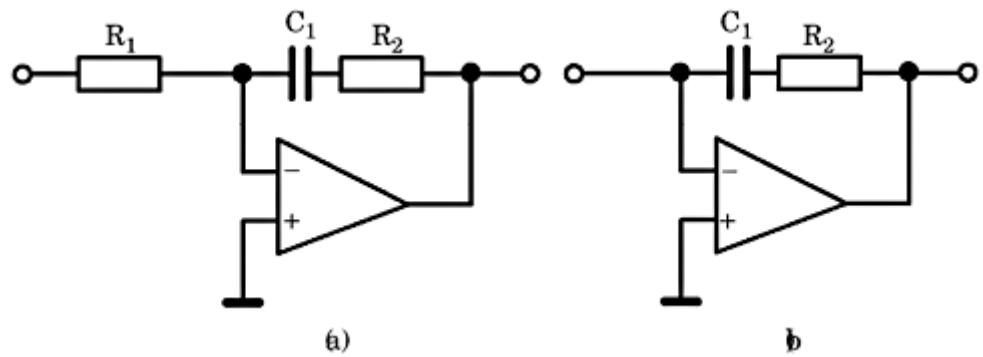
### Type 3: Active PI filters

Finally, Fig. 2.21 shows another active low-pass filter, which is commonly referred to as a PI filter.

In Fig. 2.21a, the version for voltage drive is shown; Fig. 2.21b is intended for current drive. This filter is a lead-lag filter as well. The term PI is taken from control theory, where it stands for “proportional + integral” action. For the voltage-driven PI filter, the transfer function is



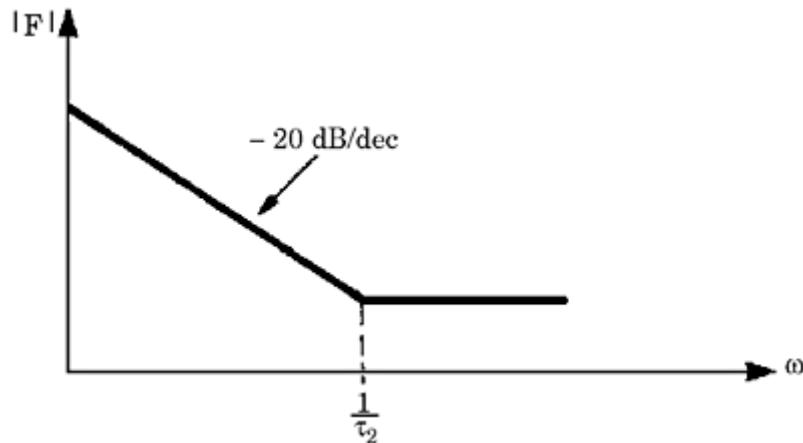
where again  $\tau_1 = R_1 C_1$  and  $\tau_2 = R_2 C_1$ . The PI filter has a pole at  $s = 0$  and therefore behaves like an integrator. It has—at least theoretically—infinite gain at



**Figure 2.21** An active PI loop filter. (a) The version for voltage drive. (b) The version for current drive.

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.



**Figure 2.22** The amplitude response of the active PI loop filter.

zero frequency. For the current-driven PI filter, the transfer function is given by

$$F(s) = \frac{1 + s\tau_2}{sC_1} \quad (2.31b)$$

Note that the inversion of polarity has been discarded in Eqs. (2.31a) and (2.31b).

The amplitude response of the active PI filter is depicted in Fig. 2.22. The Bode plot applies for both voltage-and current-driven filters.

## Controlled Oscillators

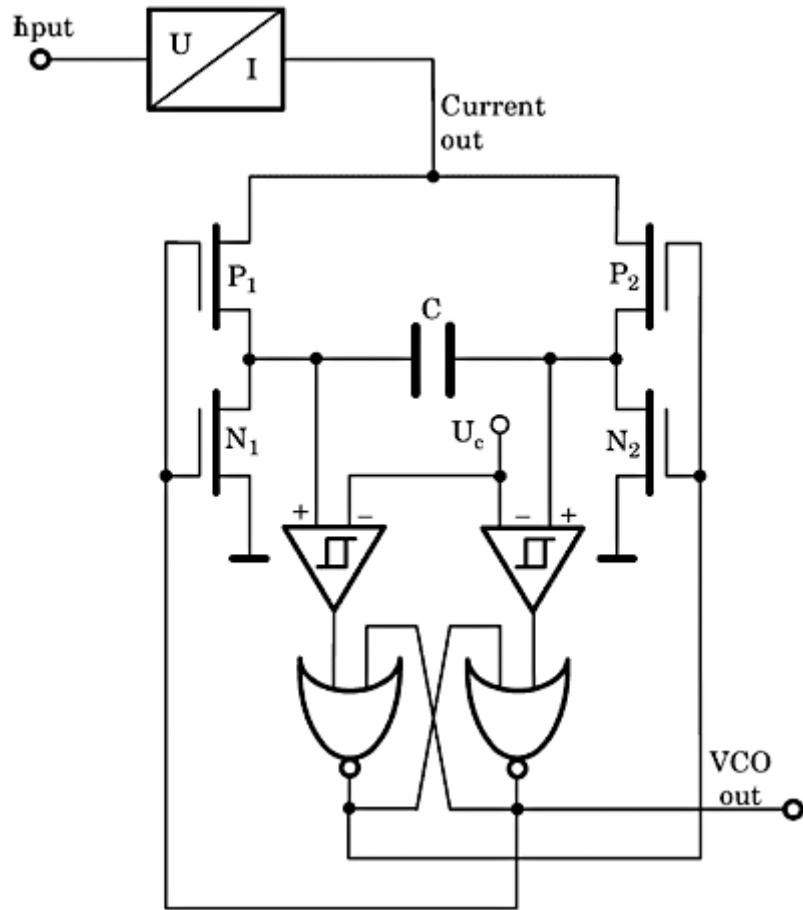
In PLLs, two fundamentally different types of controlled oscillators are used: relaxation oscillators and resonant oscillators. Relaxation oscillators are found in many PLL ICs built from standard CMOS technology. They cover a relatively restricted range of frequencies—from 0 to about 50 MHz. For higher-frequency applications, up to the microwave region, oscillators with resonant tank circuits come into play. Some of these are implemented with discrete transistors and discrete passive components; others are built as microwave-integrated circuits. We will deal with both types of oscillators in the next two sections (Secs. 2.6.1 and 2.6.2).

### Relaxation oscillators

Figure 2.23 shows the simplified schematic of a VCO, which is found in the popular digital PLL IC 74HC/HCT4046.

The operation of this circuit is as follows. First, the control signal (input) is converted into a current signal. The cross-coupled NOR gates form an RS latch. Assume that the output signal of the left NOR gate is  $H$  (high) and the output signal of the right NOR gate is  $L$  (low). Consequently,  $P$ -channel MOS transistor  $P1$  is ON, and  $N$ -channel MOS transistor  $N1$  is off; furthermore,  $P2$  is OFF,

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.



**Figure 2.23** The simplified schematic of a VCO.

and  $N_2$  is ON. Therefore, the right terminal of capacitor  $C$  is grounded, and the output current of the voltage-current converter flows into the left terminal of  $C$ ; thus, the voltage at that terminal ramps up in a positive direction. The upper threshold of the Schmitt triggers (drawn by a triangle with a hysteresis symbol in its center) is set at half the supply voltage  $U_{DD}/2$ .

When the voltage at the left side of  $C$  exceeds that threshold, the RS latch changes state. Now the left terminal of  $C$  becomes grounded, and the output current of the voltage-current converter flows into the right terminal of  $C$ . The voltage at the right side of  $C$  ramps up now, until the right Schmitt trigger switches to the  $H$  state. This process repeats infinitely. If we measured the voltage across the capacitor differentially, we would observe a triangular waveform.

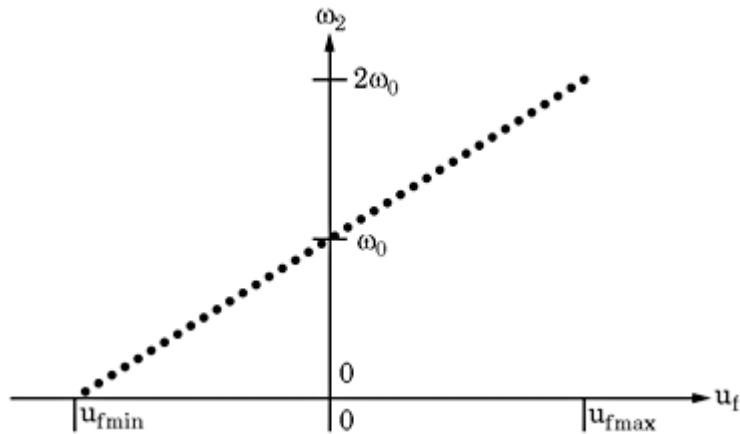
The radian frequency  $\omega_2$  of the VCO output signal is proportional to the control signal  $u_f$  and is given by

$$\omega_2 = \omega_0 + K_0 u_f \quad (2.29)$$

$K_0$  is called VCO gain, and its units are  $\text{rad s}^{-1}\text{V}^{-1}$ . The unit rad is often omitted because it is a dimensionless quantity.  $\omega_0$  is the (radian) center frequency of the PLL.

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.



**Figure 2.24** A characteristic of a VCO,  $\omega_2$  versus  $u_f$

[Figure 2.24](#) shows an idealized characteristic ( $\omega_2$  versus  $u_f$ ) of a VCO. It is assumed that the range of the control signal is symmetrical around  $u_f = 0$ . For this ideal VCO, the output frequency would be 0 for  $u_f = u_{f\min}$  and  $2\omega_0$  for  $u_f = u_{f\max}$ . Practical VCOs behave differently, however. First of all, most VCOs are powered from a unipolar power supply. Assuming the supply voltage is  $U_{DD}$ , the range of  $u_f$  must be within 0 to  $U_{DD}$ . Real VCOs operate at their center frequency when the control signal is at half the supply voltage—that is,  $u_f = U_{DD}/2$ . To be mathematically correct, for unipolar power supplies [Eq.\(2.29\)](#) should read

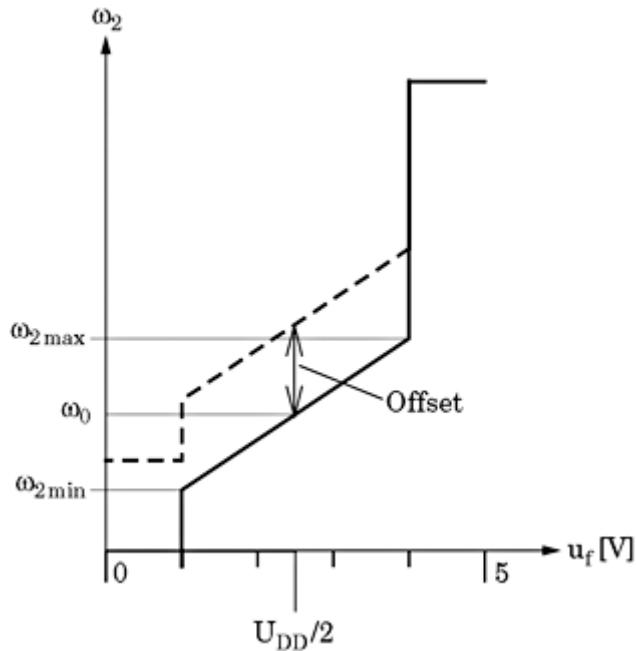
$$\omega_2 = \omega_0 + K_0(u_f - U_{DD}/2)$$

In the following, we discard that offset. Whenever we state  $u_f = 0$ , we understand that  $u_f$  is half the supply voltage.

Practical VCOs show still another limitation. Their output frequency does not vary in proportion to the control signal over its full range from 0 to  $U_{DD}$ , but this range is rather restricted between a lower and an upper threshold. For a typical VCO implemented in CMOS technology with  $U_{DD} = 5$  V, the lower threshold is around 1 V, and the upper is around 4 V. Between these thresholds, the characteristic curve is linear. Below the lower threshold, the VCO behavior is unpredictable. In the case of the 74HC/HCT4046, for example, the output frequency drops to a value near zero. Above the upper threshold, the VCO creates a very high frequency, which is independent of the control signal.

The designer of a PLL system must determine two VCO parameters, the center frequency  $\omega_0$  and the VCO gain  $K_0$ . In practical VCO circuits, these parameters are set by external components—that is, by resistors and capacitors. [Figure 2.25](#) demonstrates how this is done in the popular PLL IC of type 74HC/ HCT4046. The supply voltage is chosen  $U_{DD} = 5$  V in this example. The 74HC/HCT4046 uses three external components to set the parameters of the VCO—in other words, one capacitor  $C$  and two resistors  $R_1$  and  $R_2$ . The resistors are not shown in the schematic of [Fig. 2.24](#). Let us first consider the case

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.



**Figure 2.25** Characteristics of a typical VCO.

where  $R_2$  is chosen as infinite—meaning  $R_2$  is an open circuit. The center radian frequency  $\omega_0$  of the VCO is determined by the product  $R_1C$  (cf. the solid transfer curve in Fig. 2.25). For a supply voltage of  $U_{DD} = 5$  V, we read from the data sheet that the radian center frequency is approximately given by

$$\omega_0 \approx \frac{25}{R_1 C}$$

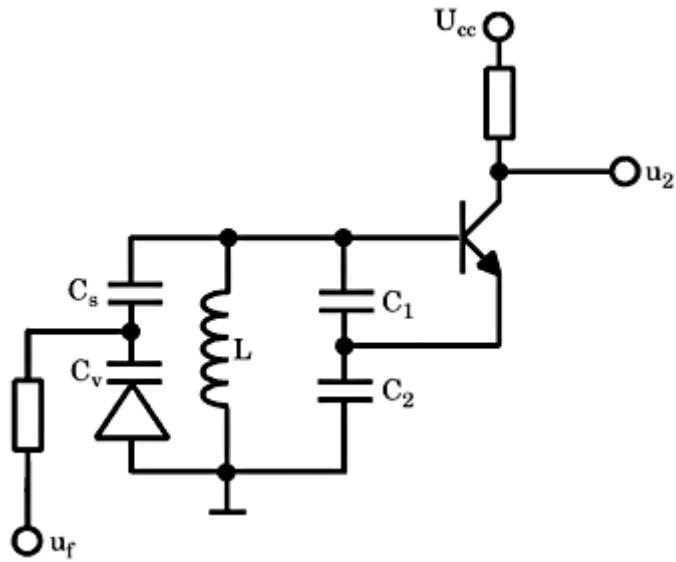
When the product  $R_1C$  is specified, the upper and lower limits of the VCO output frequency are set automatically (cf.  $\omega_{2\min}$  and  $\omega_{2\max}$  in Fig. 2.20). We conclude therefore that the VCO gain, which is simply the slope of the solid curve, is automatically fixed as soon as the product  $R_1C$  has been specified. However, we wanted to specify  $K_0$  independently of center frequency  $\omega_0$ . This is accomplished by adding another resistor  $R_2$ . This resistor introduces an offset. For a supply voltage of  $U_{DD} = 5$  V, this offset is approximately given by

$$\Delta\omega_{\text{offset}} \approx \frac{50}{R_2 C}$$

Using a resistor  $R_2$  that has a finite value (for instance, 10 kΩ) shifts the transfer curve in Fig. 2.25 upwards (cf. the dashed line). Hence, by using two resistors we are in a position to fix  $\omega_0$  and  $K_0$  independently of each other.

Because the VCO is an analog circuit, its parameters (for instance,  $\omega_0$ ) are subject to parts

spread, temperature drift, and aging. Most data sheets specify the temperature coefficient of  $\omega_0$ , which is normally given as a number of ppm (parts per million) per degree centigrade.



**Figure 2.26** Schematic of an LC oscillator with a varactor diode for tuning.

## Resonant oscillators

[Figure 2.26](#) shows the simplified schematic of a resonant oscillator. The resonant frequency is determined by the LC tank circuit.

It is tuned by means of a varactor diode. The varactor diode is reverse biased by voltage  $u_f$ , which is the output of a loop filter. The capacitance of the varactor varies with the applied reserve voltage. The tuning range is determined by the capacitance range of the varactor diode.

When the VCO is used at relatively low frequencies (up to around 100 MHz), a quartz crystal often replaces the LC tank circuit. Such VCOs are referred to as a VCXO (voltage-controlled crystal oscillator). Inductors are a critical component in the design of high-frequency oscillators. At frequencies above 1 GHz, the  $Q$  factor of discrete inductors becomes poor and is around 2 to 3. Better  $Q$  factors can be realized when the LC tank is replaced by a stripline stub. Microstrips are used in most applications; they can also be easily implemented in microwave ICs. An alternative to microstrips are radial stub lines.<sup>53</sup>

Detailed information on the design of resonant oscillators is found, for example, in the publications of *Rohde*.<sup>11, 48, 49</sup>

## Down Scalers

Down scalers (frequency dividers) come into play when the PLL is used as a frequency synthesizer. As shown in [Fig. 2.1](#), the down scaler divides the output frequency created by the VCO by a factor  $N$ , which is programmable in most cases. Down scalers are usually built from a cascade of flipflops (for instance, RS-flipflops, JK-flipflops, or toggle flipflops). One single JK-flipflop scales down the frequency applied to its clock input by 2. Two cascaded JK-flipflops scale down that frequency by 4, and so on. Arbitrary scaling factors (in other words, scaling factors that are not an integer power of 2) can be realized by adding gates to the counting circuit. This has been explained in great detail by *Rohde* (for example, Refs. [10](#) and [48](#)) and will not be discussed further here.

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

It is obvious that a counter can scale down by an integer number only—for instance, by a factor of 5 or 6. Certainly a counter cannot divide by say 5.3. Nevertheless, there exist so-called fractional-N frequency synthesizers. These are synthesizers that are indeed capable of creating, for example, an output frequency that is 5.3 times the reference frequency. Of course, the down scalers used in such synthesizers are not able to divide immediately by 5.3. Fractional-N ratios are realized by switching the divider ratio of a programmable counter from one value to another in a consecutive reference cycle. This will be discussed in more detail in [Sec. 7.1](#).

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

# Mixed-Signal PLL Analysis

## PLL Performance in the Locked State

Having discussed the components of a mixed signal PLL system, we will now analyze the dynamic performance of this type of PLL. The symbols used in the following have been defined in [Fig. 2.1](#).

If we assume that the PLL has locked and will stay locked in the near future, we can develop a linear mathematical model for the system. As will be shown in [Sec. 3.3](#), the mathematical model is used to calculate a phase-transfer function  $H(s)$ , which relates the phase  $\theta_1$  of the input signal to the phase  $\theta_2'$  of the output signal (of the down scaler):

$$H(s) = \frac{\Theta_2'(s)}{\Theta_1(s)} \quad (3.1)$$

where  $\Theta_1(s)$  and  $\Theta_2'(s)$  are the Laplace transforms of the phase signals  $\theta_1(t)$  and  $\theta_2'(t)$ , respectively. (Note that we are using lowercase symbols for time functions and uppercase symbols for their Laplace transforms throughout the text; this also applies to Greek letters. Furthermore, the symbol  $\Theta_2'(s)$  is used for the Laplace transform of phase  $\theta_2'$ .)  $H(s)$  is called *phase-transfer function*. To get an expression for  $H(s)$ , we must know the transfer functions of the individual building blocks in [Fig. 2.1](#). This transfer function will be calculated from a mathematical model that will be derived in [Sec. 3.2](#).

## The Mathematical Model for the Locked State

As derived in [Sec. 2.4](#), in the locked state the output signal  $u_d$  of the phase detector can be approximated by

$$u_d \approx K_d \theta_e$$

for phase detectors with voltage output or

$$i_d \approx K_P \theta_e$$

for phase detectors with current output,

hence the mathematical model of the phase detector is simply a zero-order block with gain  $K_d$  or  $K_P$  (also referred to as “gain block”). The transfer function of the phase detector is therefore given by

$$\frac{U_d(s)}{\Theta_e(s)} = K_d \quad (\text{voltage output}) \quad (3.2)$$

$$\frac{I_d(s)}{\Theta_e(s)} = K_P \quad (\text{current output}) \quad (3.3)$$

where  $s$  is the Laplace operator. ([App. B](#) provides an introduction to Laplace transforms.)

The transfer functions of the loop filter have already been derived in [Sec. 2.5](#) and are denoted here with  $F(s)$ . Let’s now look at the transfer function of the VCO. As stated in [Eq. \(1.1\)](#), the angular frequency of the VCO is given by

$$\omega_2(t) = \omega_0 + \Delta\omega_2(t) = \omega_0 + K_0 \cdot u_f(t) \quad (3.4)$$

where  $K_0$  is called VCO gain (dimension: rad s<sup>-1</sup>V<sup>-1</sup>). The model of the VCO should yield the output phase  $\theta_2$ , however, and not the output frequency  $\omega_2$ . By definition, the phase  $\theta_2$  is given by the integral over the frequency variation  $\Delta\omega_2$

$$\theta_2(t) = \int \Delta\omega_2(t) dt = K_0 \int u_f(t) dt \quad (3.5a)$$

In the Laplace transform, integration over time corresponds to division by  $s$ , so the Laplace transform of the output phase  $\theta_2$  is given by

$$\Theta_2(s) = \frac{K_0}{s} U_f(s) \quad (3.5b)$$

The transfer function of the VCO is therefore given by

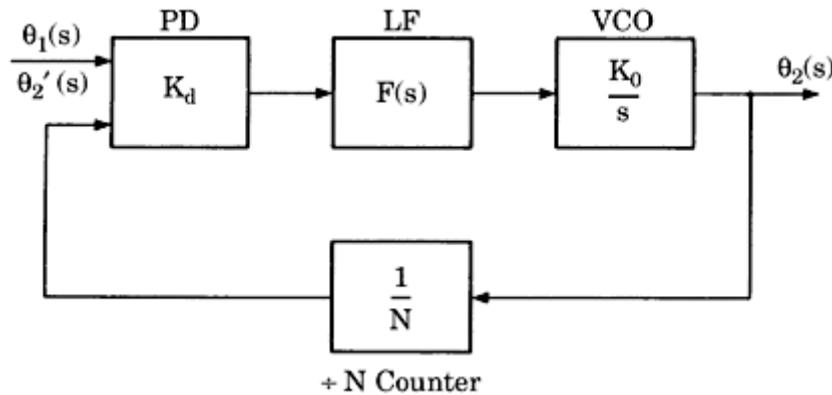
$$\frac{\Theta_2(s)}{U_f(s)} = \frac{K_0}{s} \quad (3.6)$$

For phase signals, the VCO simply represents an integrator.

Lastly, we must also know the mathematical model of the (optional) divide-by- $N$  counter (cf. [Fig. 2.1](#)). Because this counter divides the frequency created by the VCO by a

factor  $N$ , it also scales down the output phase of the VCO by  $N$ . Hence, the down scaler is nothing more than a “gain block” having gain  $1/N$ .

We are now in the position to draw the mathematical model of the locked state (see [Fig. 3.1](#)).



**Figure 3.1** The mathematical model for the locked state of the PLL.

Based on this model, we are going now to derive a number of transfer functions and related parameters (cf. [Sec. 3.3](#)).

## A Definition of Transfer Functions

The model in [Fig. 3.1](#) enables us to analyze the tracking performance of the PLL—in other words, the ability of the system to maintain phase tracking when excited by phase steps, frequency steps, or other excitation signals. Because phase detectors with voltage output don't have the same mathematical model as phase detectors with current output, the analysis is done separately for the two categories of phase detectors.

### The PLL transfer function for systems using the voltage output phase detector

From the model in [Fig. 3.1](#), the phase-transfer function  $H(s)$  is computed. We get

$$H(s) = \frac{\Theta_2'(s)}{\Theta_1(s)} = \frac{K_0 K_d F(s)/N}{s + K_0 K_d F(s)/N} \quad (3.7)$$

In addition to the phase-transfer function, an error-transfer function  $H_e(s)$  has been defined.  $H_e(s)$  is defined by

$$H_e(s) = \frac{\Theta_e(s)}{\Theta_1(s)} = \frac{s}{s + K_0 K_d F(s)/N} \quad (3.8)$$

$H_e(s)$  relates phase error  $\theta_e$  to the input phase  $\theta_1$ . Between  $H_e(s)$  and  $H(s)$ , we have the simple relation

$$H_e(s) = 1 - H(s) \quad (3.9)$$

To analyze the phase-transfer function, we have to insert the loop filter transfer function

[Eqs. (2.29a), (2.30a), (2.31a)] into Eq. (3.7). For the three different loop filters (Figs. 2.17a, 2.19a, and 2.21a), we get

■ For the passive lead-lag filter

$$H(s) = \frac{\frac{K_0 K_d}{N} \frac{1 + s\tau_2}{\tau_1 + \tau_2}}{s^2 + s \frac{1 + K_0 K_d \tau_2 / N}{\tau_1 + \tau_2} + \frac{K_0 K_d / N}{\tau_1 + \tau_2}} \quad (3.10)$$

■ For the active lead-lag filter

$$H(s) = \frac{\frac{K_0 K_d K_a}{N} \frac{1 + s\tau_2}{\tau_1}}{s^2 + s \frac{1 + K_0 K_d K_a \tau_2 / N}{\tau_1} + \frac{K_0 K_d K_a / N}{\tau_1}} \quad (3.11)$$

■ For the active PI filter

$$H(s) = \frac{\frac{K_0 K_d}{N} \frac{1 + s\tau_2}{\tau_1}}{s^2 + s \frac{K_0 K_d \tau_2 / N}{\tau_1} + \frac{K_0 K_d / N}{\tau_1}} \quad (3.12)$$

In circuit and control theory it is common practice to write the denominator of the transfer function in the so-called normalized form<sup>3</sup>

$$\text{Denominator} = s^2 + 2\zeta\omega_n s + \omega_n^2$$

where  $\omega_n$  is the natural frequency and  $\zeta$  is the damping factor. The denominator of Eqs. (3.10) to (3.12) will take this form if the following substitutions are made:

■ For the passive lead-lag filter:

$$\omega_n = \sqrt{\frac{K_0 K_d}{N(\tau_1 + \tau_2)}}, \quad \zeta = \frac{\omega_n}{2} \left( \tau_2 + \frac{N}{K_0 K_d} \right) \quad (3.13)$$

■ For the active lead-lag filter:

$$\omega_n = \sqrt{\frac{K_0 K_d K_a}{N \tau_1}}, \quad \zeta = \frac{\omega_n}{2} \left( \tau_2 + \frac{N}{K_0 K_d K_a} \right) \quad (3.14)$$

■ For active PI filter:

(3.15)

$$\omega_n = \sqrt{\frac{K_0 K_d}{N\tau_1}}, \quad \zeta = \frac{\omega_n}{2}\tau_2$$

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

(The natural frequency  $\omega_n$  must never be confused with the center frequency  $\omega_0$  of the PLL.) Inserting these substitutions into Eqs. (3.10) through (3.12), we get the following phase-transfer functions:

■ For the passive lead-lag filter:

$$H(s) = \frac{s\omega_n \left( 2\zeta - \frac{\omega_n}{K_0 K_d / N} \right) + \omega_n^2}{s^2 + 2s\zeta\omega_n + \omega_n^2} \quad (3.16)$$

■ For the active lead-lag filter:

$$H(s) = \frac{s\omega_n \left( 2\zeta - \frac{\omega_n}{K_0 K_d K_a / N} \right) + \omega_n^2}{s^2 + 2s\zeta\omega_n + \omega_n^2} \quad (3.17)$$

■ For the active PI filter:

$$H(s) = \frac{2s\zeta\omega_n + \omega_n^2}{s^2 + 2s\zeta\omega_n + \omega_n^2} \quad (3.18)$$

Aside from the parameters  $\omega_n$  and  $\zeta$ , only the parameters  $K_d$ ,  $K_0$ ,  $K_a$ , and  $N$  appear in Eqs. (3.16) to (3.18). The term  $K_d K_0 / N$  in Eqs. (3.16) and (3.18) is called *loop gain* and has the dimension of angular frequency ( $\text{rad s}^{-1}$ ). In Eq. (3.17), the term  $K_d K_0 K_a / N$  is called *loop gain*. If the condition

$$K_d K_0 / N \gg \omega_n \text{ or } K_d K_0 K_a / N \gg \omega_n$$

is true, the PLL system is said to be a *high-gain loop*. If the reverse is true, the system is called a *low-gain loop*. Most practical PLLs are high-gain loops. For high-gain loops, Eqs. (3.16) to (3.18) become approximately identical and read

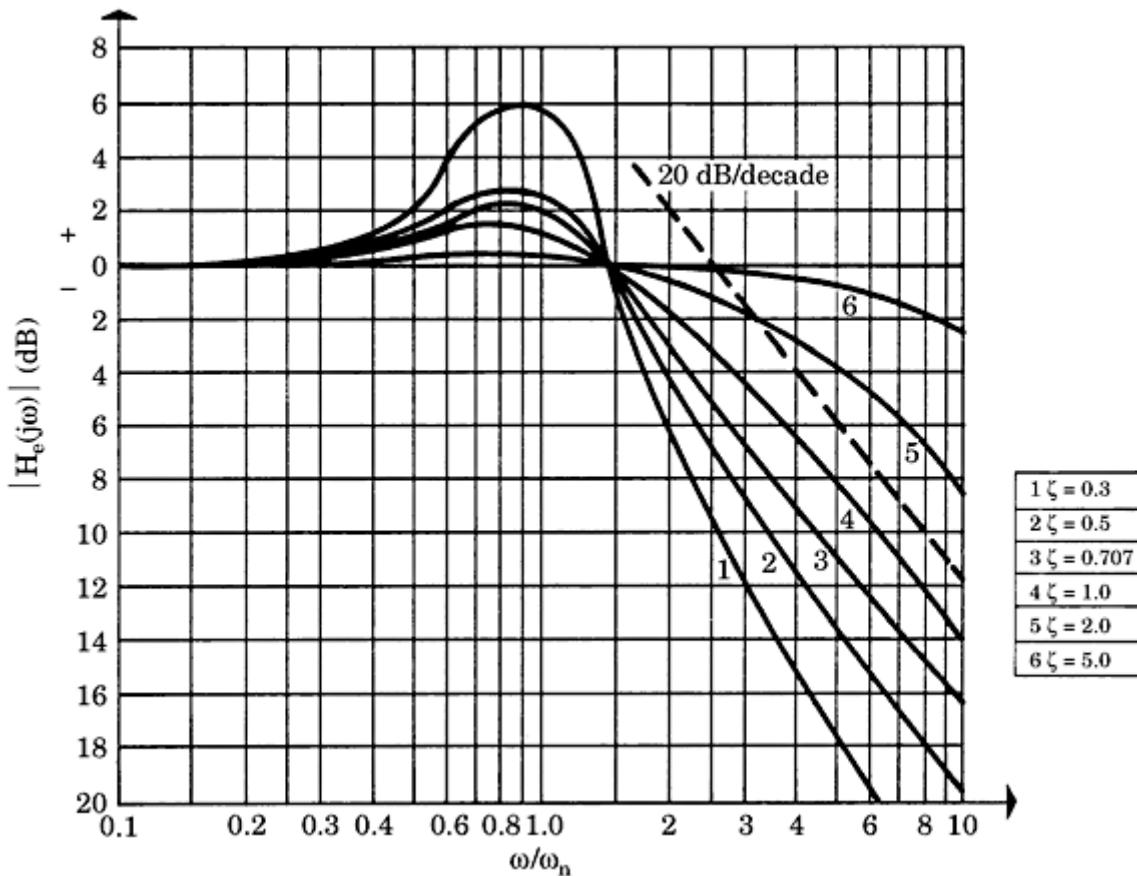
$$H(s) \approx \frac{2s\zeta\omega_n + \omega_n^2}{s^2 + 2s\zeta\omega_n + \omega_n^2} \quad (3.19)$$

for all filters considered hitherto. Similarly, assuming a high-gain loop, we get for the error-transfer function  $H_e(s)$  for all three filter types the approximate expression

$$H_e(s) \approx \frac{s^2}{s^2 + 2s\zeta\omega_n + \omega_n^2} \quad (3.20)$$

To investigate the transient response of a control system, it is customary to plot a Bode

diagram of its transfer function. The Bode diagram of the phase-transfer function is obtained by putting  $s = j\omega$  in Eq. (3.19) and plotting the magnitude



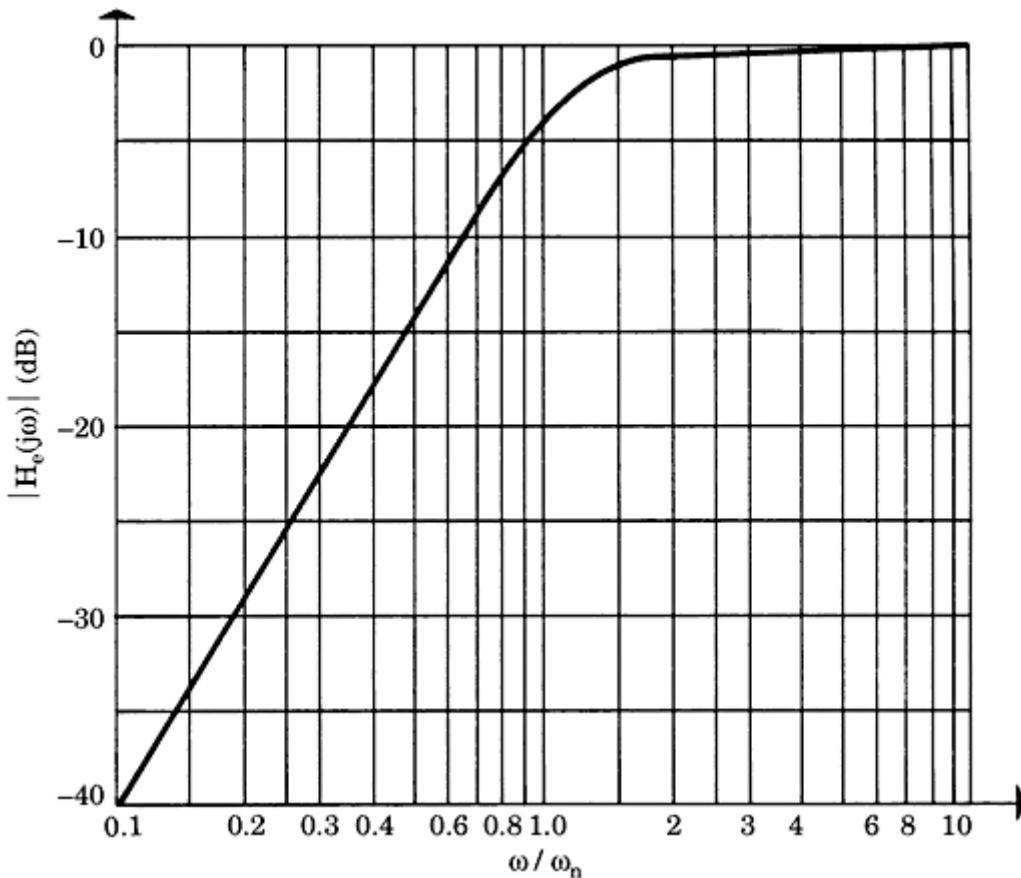
**Figure 3.2** A Bode diagram of the phase-transfer function  $H(\omega)$ . (Adapted from Gardner<sup>1</sup> with permission.)

(absolute value)  $|H(\omega)|$  as a function of angular frequency  $\omega$  (Fig. 3.2). Both scales are usually logarithmic. The frequency scale is further normalized to the natural frequency  $\omega_n$ . Thus, the graph is valid for every second-order PLL system.

We can see from Fig. 3.2 that the second-order PLL is actually a low-pass filter for input phase signals  $\theta_1(t)$ , whose frequency spectrum is flat between zero and approximately the natural frequency  $\omega_n$ . This means the second-order PLL is able to track for phase and frequency modulations of the reference signal as long as the modulation frequencies remain within an angular frequency band that's roughly between zero and  $\omega_n$ .

The damping factor  $\zeta$  has an important influence on the dynamic performance of the PLL. For  $\zeta = 1$ , the system is critically damped. If  $\zeta$  is made smaller than unity, the transient response becomes oscillatory; the smaller the damping factor, the larger becomes the overshoot. In most practical systems, an optimally flat frequency-transfer function is the goal. The transfer function is optimally flat for  $\zeta = 1/\sqrt{2}$ , which corresponds to a second-order Butterworth low-pass filter. If  $\zeta$  is made considerably larger than unity, the transfer function flattens out, and the dynamic response becomes sluggish.

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.



**Figure 3.3** A Bode diagram of the error-transfer function  $H_e(\omega)$ .

A Bode plot of  $H_e(s)$  is shown in Fig. 3.3. The value of 0.707 has been chosen for  $\zeta$ . The diagram shows that for modulation frequencies smaller than the natural frequency  $\omega_n$ , the phase error remains relatively small. For larger frequencies, however, the phase error  $\theta_e$  becomes as large as the reference phase  $\theta_1$ , which means the PLL is no longer able to maintain phase tracking.

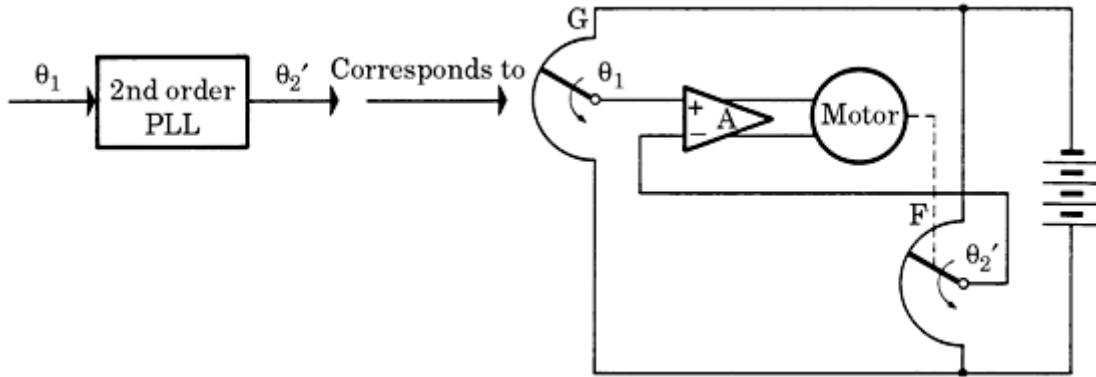
As in amplifiers, the bandwidth of a PLL is often specified by the 3-dB corner frequency  $\omega_{3db}$ . This is the radian frequency where the closed loop gain has dropped by 3 dB, which is referred to the closed loop gain at DC.  $\omega_{3db}$  is given by

$$\omega_{3db} = \omega_n [1 + 2\zeta^2 + \sqrt{(1 + 2\zeta^2)^2 + 1}]^{1/2} \quad (3.21)$$

For a damping factor  $\zeta = 0.7$ ,  $\omega_{3db}$  becomes  $\omega_{3db} = 2.06 \omega_n$ , which is about twice the natural frequency.

Knowing that a second-order PLL in the locked state behaves very much like a servo or follow-up control system, we can plot a simple model for the locked PLL (Fig. 3.4). The model consists of a reference potentiometer  $G$ , a servo amplifier, and a follow-up

potentiometer  $F$  whose shaft is driven by an electric motor. In this model, the reference phase  $\theta_1$  is represented by the shaft position of the reference potentiometer  $G$ . The phase of the output signal of the VCO  $\theta_2'$  ( $t$ ) is represented by the shaft position of the follow-up potentiometer. If the



**Figure 3.4** A simple electromechanical analogy of the linearized second-order PLL. In this servo system, the angles  $\theta_1$  and  $\theta_2'$  correspond to the phases  $\theta_1$  and  $\theta_2'$ , respectively, of the PLL system.

shaft position of the reference potentiometer is varied slowly, the servo system will be able to maintain tracking of the follow-up potentiometer. If  $\theta_1(t)$  is changed too abruptly, the servo system will lose tracking and thus large phase errors  $\theta_e$  will result.

So far we have seen that a linear model is best suited to explain the tracking performance of the PLL if it is assumed the PLL is initially locked. If the PLL is initially unlocked, however, the phase error  $\theta_e$  can take on arbitrarily large values, and the linear model is no longer valid. When we try to calculate the acquisition process of the PLL itself, we must use a model that also accounts for the nonlinear effect of the phase detector. This will be dealt with in [Sec. 3.8](#).

Knowing the phase-transfer function  $H(s)$  and the error-transfer function  $H_e(s)$  of the PLL, we can calculate its response on the most important excitation signals. This will be done in [Sec. 3.4](#).

### The PLL transfer function for systems using current output phase detector

To compute the phase-transfer function of the PLL using a current output phase detector, we can apply the model in [Fig. 3.1](#) again. We only have to replace detector gain  $K_d$  by  $K_P$  [as defined in [Eq. \(2.27\)](#)] and insert the appropriate filter transfer function into block  $F(s)$ . Because the passive lead-lag loop filter is almost exclusively used in PLLs working with charge pump PDs, we will apply the transfer function given in [Eq. \(2.29b\)](#). For  $H(s)$ , we then get

$$H(s) = \frac{K_P K_0 (1 + s\tau_2)}{s^2 N C_1 + K_P K_0 s \tau_2 + K_P K_0} \quad (3.22)$$

where  $\tau_2 = R_2 C_1$  (cf. [Fig. 2.17b](#)). When making the substitutions

$$\omega_n^2 = \frac{K_P K_0}{N C_1}, \quad \zeta = \frac{\omega_n \tau_2}{2} \quad (3.23)$$

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

we get the normalized form

$$H(s) = \frac{2s\zeta\omega_n + \omega_n^2}{s^2 + 2s\zeta\omega_n + \omega_n^2} \quad (3.24)$$

This is identical to the phase-transfer function we obtained for the PLL using a phase detector with voltage output [cf. [Eq. \(3.18\)](#)].

## Transient Response of the PLL in the Locked State

Knowing the phase-transfer function  $H(s)$  and the error-transfer function  $H_e(s)$  of the PLL, we can calculate its response on the most important excitation signals. We therefore analyze the PLL's answer to

- A phase step
- A frequency step
- A frequency ramp

applied to its reference input.

### Phase step applied to the reference input

A reference signal performing a phase step at time  $t = 0$  has been shown in [Fig. 2.2a](#). In this case, the phase signal  $\theta_1(t)$  is a step function,

$$\theta_1(t) = u(t) \cdot \Delta\Phi \quad (3.25)$$

where  $u(t)$  is the unit step function and  $\Delta\Phi$  is the size of the phase step. For the Laplace transform  $\Theta_1(s)$ , we therefore get

$$\Theta_1(s) = \Delta\Phi/s \quad (3.26)$$

The phase error  $\theta_e$  is obtained from

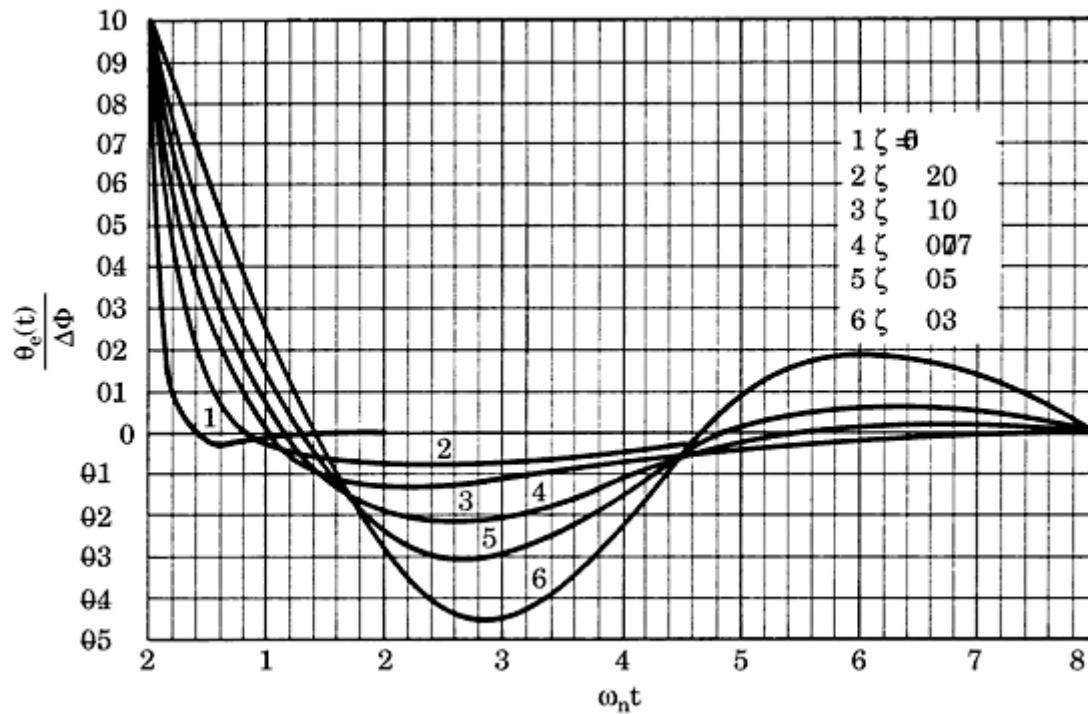
$$\Theta_e(s) = H_e(s)\Theta_1(s) = H_e(s) \cdot \Delta\Phi/s \quad (3.27)$$

Inserting [Eq. \(3.20\)](#) into [Eq. \(3.27\)](#) yields

$$\Theta_e(s) = \frac{\Delta\Phi}{s} \frac{s^2}{s^2 + 2s\zeta\omega_n + \omega_n^2} \quad (3.28)$$

Applying the inverse Laplace transform to [Eq. \(3.28\)](#), we get the phase error functions  $\theta_e(t)$  shown in [Fig. 3.5](#). The phase error has been normalized to the phase step  $\Delta\Phi$ .

Phase error functions have been plotted for different damping factors. The initial phase error  $\theta_e(0)$  equals  $\Delta\Phi$  or any value of  $\zeta$ . For  $t \rightarrow \infty$ , the phase error  $\theta_e(\infty)$  approaches zero.



**Figure 3.5** Transient response of a linear second-order PLL to a phase step  $\Delta\Phi$  applied at  $t = 0$ . The PLL is assumed to be a high-gain loop. (*Adapted from Gardner*<sup>L</sup> with permission.)

This can also be shown by the final value theorem of the Laplace transform,<sup>3</sup> which reads

$$\theta_e(\infty) = \lim_{s \rightarrow 0} s\Theta_e(s) = 0 \quad (3.29)$$

### Frequency step applied to the reference input

If a frequency step is applied to the input of the PLL, the angular frequency of the reference signal becomes

$$\omega_1(t) = \omega_0' + \Delta\omega \cdot u(t) \quad (3.30)$$

where  $\Delta\omega$  is the magnitude of the frequency step. Because the phase  $\theta_1(t)$  is the integral over the frequency variation  $\Delta\omega$ , we have

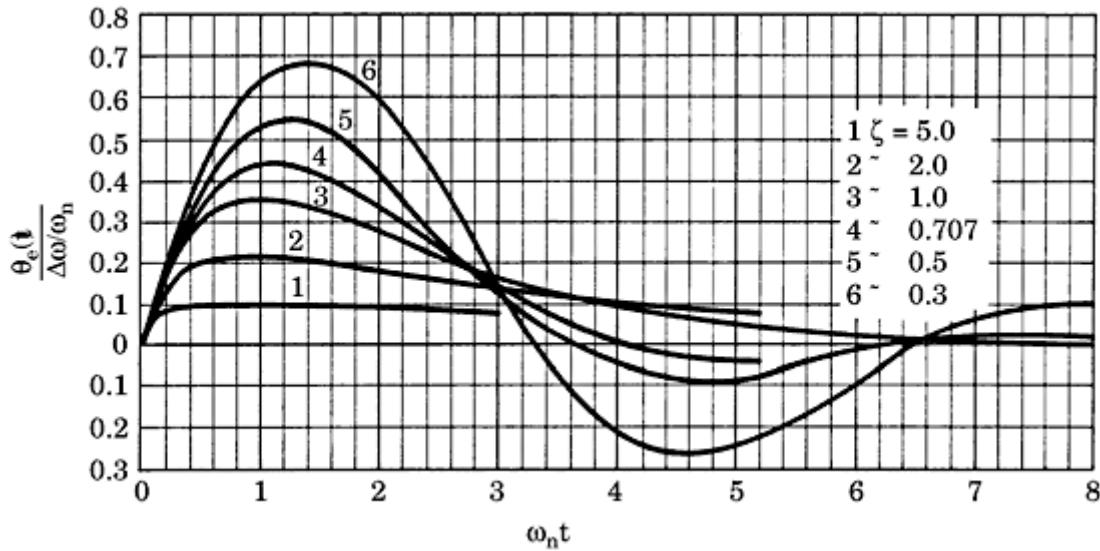
$$\theta_1(t) = \Delta\omega \cdot t \quad (3.31)$$

In other words, the phase is a ramp function now. For the Laplace transform  $\Theta_1(s)$  we get therefore

$$\Theta_1(s) = \Delta\omega/s^2 \quad (3.32)$$

Performing the same calculation as the preceding one, we get for the phase error

$$\Theta_e(s) = \frac{\Delta\omega}{s^2} \frac{s^2}{s^2 + 2s\zeta\omega_n + \omega_n^2} \quad (3.33)$$



**Figure 3.6** The transient response of a linear second-order PLL to a frequency step  $\Delta\omega$  applied to its reference input at  $t = 0$ . (Adapted from Gardner<sup>L</sup> with permission.)

Applying the inverse Laplace transform to Eq. (3.33), we get the phase error curves shown in Fig. 3.6. If we again apply the final value theorem to Eq. (3.29), it turns out the phase error approaches 0 when  $t \rightarrow \infty$ . This is only true, however, for high-gain loops. For low-gain loops, the numerator of Eq. (3.33) would have an additional first-order term in  $s$ ; hence, the phase error  $\theta_e(\infty)$  becomes nonzero.

### Frequency ramp applied to the reference input

If a frequency ramp is applied to the PLL's reference input, the angular frequency  $\omega_1$  is given by

$$\omega_1(t) = \omega_0' + \Delta\dot{\omega} \cdot t \quad (3.34)$$

where  $\Delta\dot{\omega}$  is the rate of change of the reference frequency. Because the phase  $\theta_1(t)$  is the integral over the frequency variation, we get

$$\theta_1(t) = \Delta\dot{\omega} \frac{t^2}{2} \quad (3.35)$$

The Laplace transform  $\Theta_1(s)$  now becomes

$$\Theta_1(s) = \frac{\Delta\dot{\omega}}{s^3} \quad (3.36)$$

Assuming a high-gain loop again and applying the final value theorem of the Laplace

transform, the final phase error  $\theta_e(\infty)$  is

$$\theta_e(\infty) = \lim_{s \rightarrow 0} sH_e(s)\Theta_1(s) = \frac{\Delta\dot{\omega}}{\omega_n^2} \quad (3.37)$$

Now we remember that our linear model is valid for small phase errors only. Because for large phase error the output signal of the phase detector is not

proportional to phase error but to the sine of phase error, the last equation can be shown to read actually<sup>1</sup>

$$\sin \theta_e(\infty) = \frac{\Delta \dot{\omega}}{\omega_n^2} \quad (3.38)$$

Because the sine function cannot exceed unity, the maximum rate of change of the reference frequency that does not cause lock-out is

$$\Delta \dot{\omega}_{\max} = \omega_n^2 \quad (3.39)$$

This result has two consequences:

- If the reference frequency is swept at a rate larger than  $\omega_n^2$ , the system will unlock.
- If the system is initially unlocked, it cannot become locked if the reference frequency is simultaneously swept at a rate larger than  $\omega_n^2$ .

Practical experience with PLLs has shown that [Eq. \(3.39\)](#) presents a theoretical limit that is normally not practicable. If the reference frequency is swept in the presence of noise, the rate at which an initially unlocked PLL can become locked is markedly less than  $\omega_n^2$ . A more practical design limit for  $\Delta \dot{\omega}_{\max}$  is considered to be<sup>1</sup>

$$\Delta \dot{\omega}_{\max} = \frac{\omega_n^2}{2} \quad (3.40)$$

## Steady-State Error of the PLL

The following discussion assumes that a phase detector with voltage output is used. The results are also valid, however, for charge pump PDs, but in the equations some minor modifications would have to be made (for example, setting  $K_P$  instead of  $K_d$ , and so on).

In control systems, the steady-state error is defined as the deviation of the controlled variable from the setpoint after the transient response has died out. For the PLL, the steady-state error is simply the phase error  $\theta_e(\infty)$ . To see how the PLL settles on various excitation signals applied to its input, we will calculate the steady-state error for a phase step, a frequency step, and a frequency ramp. When the input phase  $\theta_1(t)$  is given, the phase error  $\theta_e(t)$  is computed from the error-transfer function  $H_e(s)$  as defined in [Eq. \(3.8\)](#). It turns out that the steady-state error depends largely on the number of “integrators” present in the control system—in other words, on the number of poles at  $s = 0$  of the open-loop transfer function. Using [Eq. \(3.8\)](#) and the

final value theorem of the Laplace transform [see [App. B](#), [Eq. \(B.19\)](#)],  $\theta_e(\infty)$  is given by

$$\theta_e(\infty) = \lim_{s \rightarrow 0} s\Theta_1(s) \frac{s}{s + K_0 K_d F(s)/N}$$

For the transfer function  $F(s)$  of the loop filter, we choose a generalized expression

$$F(s) = \frac{P(s)}{Q(s)s^M}$$

where  $P(s)$  and  $Q(s)$  can be any polynomials in  $s$ , and  $M$  is the number of poles at  $s = 0$ . For many loop filters,  $M = 0$ . For the active PI loop filter, as defined in [Eq. \(2.31a\)](#),  $M = 1$ . It is possible to build loop filters having more than one pole at  $s = 0$ , but the larger the number of integrators, the more difficult it becomes to get a stable system. In most cases,  $P(s)$  is a first-order polynomial, and  $Q(s)$  is a polynomial of order 0 or 1. When inserting  $F(s)$  into the equation for the steady state error, we get

$$\theta_e(\infty) = \lim_{s \rightarrow 0} \frac{s^2 s^M Q(s) \Theta_1(s)}{s \cdot s^M Q(s) + K_0 K_d P(s)/N}$$

Let's now calculate the steady-state error for different excitations at the reference input of the PLL.

**Case study 1.** *Phase step applied to the reference input.* If a phase step of size is applied to the reference input, we have (cf. [Sec. 3.4](#))

$$\Theta_1(s) = \frac{\Delta\Phi}{s}$$

Hence, the steady-state error becomes

$$\theta_e(\infty) = \lim_{s \rightarrow 0} \frac{s^2 s^M Q(s) \Delta\Phi}{s(s^M Q(s) + K_0 K_d P(s)/N)}$$

This quantity becomes 0 for any  $M$ —even for  $M = 0$ . Hence, the phase error settles to zero for any type of loop filter.

**Case study 2.** *Frequency step applied to the reference input.* In case of a frequency step, the phase  $\theta_1(t)$  becomes a ramp function (as shown in [Sec. 3.4](#)), and for its Laplace transform, we get

$$\Theta_1(s) = \frac{\Delta\omega}{s^2}$$

**Any use is subject to the Terms of Use as given at the website.**

where  $\Delta\omega$  is the size of the (angular) frequency step. The steady-state error is given by

$$\theta_e(\infty) = \lim_{s \rightarrow 0} \frac{s^2 s^M Q(s) \Delta\omega}{s^2 (s \cdot s^M Q(s) + K_0 K_d P(s)/N)}$$

This only becomes 0 if  $M$  is greater or equal to 1—in other words, if the loop filter has at least one pole at  $s = 0$ . Thus, the open-loop transfer function of the PLL must have at least 2 poles at  $s = 0$  if the steady-state error caused by a frequency step must become zero.

**Case study 3.** *Frequency ramp applied to the reference input.* As shown in [Sec. 3.4](#), for a frequency ramp the Laplace transform of the phase signal  $\theta_1(t)$  becomes

$$\Theta_1(s) = \frac{\Delta\dot{\omega}}{s^3}$$

where  $\Delta\dot{\omega}$  is the rate of change of angular frequency. For the steady-state error, we get

$$\theta_e(\infty) = \lim_{s \rightarrow 0} \frac{s^2 s^M Q(s) \Delta\dot{\omega}}{s^3 (s \cdot s^M Q(s) + K_0 K_d P(s)/N)}$$

Here the steady-state error only approaches zero if the loop filter has at least two poles at  $s = 0$ . For  $M = 2$  and  $Q(s) = 1$ , the order of the PLL becomes 3. Because each pole of the transfer function causes a phase shift of nearly  $90^\circ$  at higher frequencies, the overall phase shift can approach  $270^\circ$ , which means the system can become unstable. To achieve a stable loop, the poles must be compensated for by zeros—that is, the loop filter must have at least one zero, which must be correctly placed, of course.<sup>3</sup> (Appropriate placement of poles and zeros will be discussed in [Chap.9](#).)

## The Order of the PLL System

### The number of poles

Most PLLs considered hitherto were second-order PLLs. The loop filter had one pole, and the VCO had a pole at  $s = 0$ , so the whole system had two poles. Generally, the order of a PLL is always higher by 1 than the order of the loop filter. If the loop filter is omitted—in other words, if the output of the phase detector directly controls the VCO—we obtain a first-order PLL. (We will discuss the first-order PLL in [Sec. 3.6.2](#).) In PLL applications, first-order systems are rarely used, because they offer little noise suppression (more about noise in [Chap. 4](#)).

Because higher-order loop filters offer better noise cancellation, loop filters of order 2 and higher are used in critical applications. As mentioned, it is much

more difficult to obtain a stable higher-order system than a lower-order one. We will deal with higher-order loops in [Chap. 9](#).

### A special case: the first-order PLL

The first-order PLL is an extremely simple system. When we omit the loop filter,  $F(s)$  in [Eq. \(3.7\)](#) becomes 1, and for the phase-transfer function we obtain

$$H(s) = \frac{1}{1 + \frac{sN}{K_0 K_d}} \quad (3.41)$$

This is the transfer function of a first-order low-pass filter having a 3-dB angular corner frequency  $\omega_{3db} = K_0 K_d / N$ . As we will see in [Sec. 3.9.1](#), the term  $K_0 K_d / N$  is identical with the hold range of the PLL.

Because the hold range is generally much larger than the natural frequency  $\omega_n$  of second-order PLLs, the first-order PLL has large bandwidth and hence tracks phase and frequency variations of the input signal very rapidly. Due to its high bandwidth, however, the first-order PLL does not suppress noise superimposed to the input signal. Because this is an undesirable property in most PLL applications, the first-order PLL is rarely utilized here. First-order PLLs really do exist in applications where noise is not a primary concern. In [Chap. 11](#), we will deal with a first-order all-digital PLL system frequently used in FSK modems.

### PLL Performance in the Unlocked State

As we have seen in [Sec. 3.2](#), the linear model of the PLL is valid only when the PLL is in the locked state. When the PLL is out of lock, its model becomes much more complicated and is nonlinear, of course. In this section, we shall develop a mathematical model that is valid in the unlocked state of the PLL. Based on that model, we will develop a mechanical analogy that is much simpler to analyze and that will help us derive the relevant parameters describing the acquisition and lock-out processes of PLLs.

This analogy should answer the following questions:

- Under what conditions will the PLL get locked?
- How much time does the lock-in process need?
- Under what conditions will the PLL lose lock?

### Mathematical Model for the Unlocked State

The following discussion assumes that a phase detector with voltage output is used. The results are valid, however, also for charge pump PDs, but in the equations some minor modifications would have to be made (for example, setting  $K_P$  instead of  $K_d$ , and so on).

**Any use is subject to the Terms of Use as given at the website.**

The mathematical model depends somewhat on the types of phase detectors and loop filters used in a particular PLL configuration. For the following analysis, we assume the PLL contains a type 1 phase detector (multiplier) and a type 1 loop filter (passive lead-lag filter). It can be shown that the behavior of the PLL in the unlocked state is described by a nonlinear differential equation of the form<sup>4</sup>

$$\ddot{\theta}_e + \dot{\theta}_e \frac{1 + K_0 K_d \tau_2 \cos \theta_e / N}{\tau_1 + \tau_2} + \frac{K_0 K_d / N}{\tau_1 + \tau_2} \sin \theta_e = \ddot{\theta}_1 + \dot{\theta}_1 \frac{1}{\tau_1 + \tau_2} \quad (3.42)$$

This equation can be simplified. First, the substitutions of [Eq. \(3.13\)](#) are made for  $\tau_1$  and  $\tau_2$ . Next, in most practical cases, the inequality

$$\frac{1}{\tau_2} \ll K_0 K_d / N \quad (3.43)$$

holds. This leads to the simplified differential equation

$$\ddot{\theta}_e + 2\zeta\omega_n \dot{\theta}_e \cos \theta_e + \omega_n^2 \sin \theta_e = \ddot{\theta}_1 + \dot{\theta}_1 \frac{\omega_n^2}{K_0 K_d / N} \quad (3.44)$$

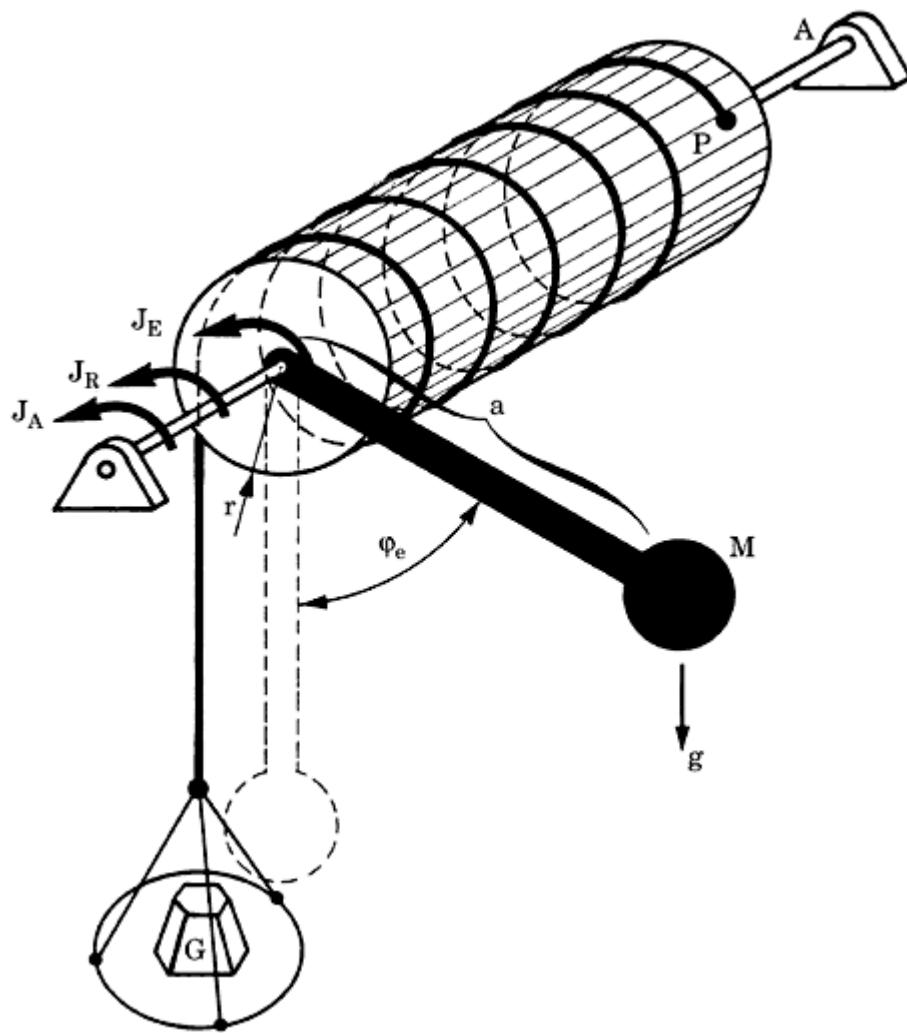
The nonlinearities in this equation stem from the trigonometric terms  $\sin \theta_e$  and  $\cos \theta_e$ . As already stated, there is no exact solution for this problem. We find, however, that [Eq. \(3.44\)](#) is almost identical to the differential equation of a somewhat special mathematical pendulum, as shown in [Fig. 3.7](#). A beam having a mass  $M$  is rigidly fixed to the shaft of a cylinder, which can rotate freely around its axis. A thin rope is attached at point  $P$  to the surface of the cylinder and is then wound several times around the latter. The outer end of the rope hangs down freely and is attached to a weighing platform. If there is no weight on the platform, the pendulum is assumed to be in a vertical position with  $\varphi_e = 0$ . If some weight  $G$  is placed on the platform, the pendulum will be deflected from its quiescent position and will eventually settle at a final deflection angle  $\varphi_e$ . The dynamic response of the pendulum can be calculated by Newton's third law

$$T\ddot{\varphi}_e = \sum_i J_i \quad (3.45)$$

where  $T$  is the moment of inertia of the pendulum plus the cylinder,  $\varphi_e$  is the angle of deflection, and  $J_i$  is the driving torque. Three different torques can be identified in the mechanical system of [Fig. 3.7](#):

- The torque  $J_E$  generated by gravitation of the mass  $M$ ;  $J_E = -Mag \sin \varphi_e$ , where  $a$  is the length of the beam and  $g$  is acceleration due to gravity.
- A friction torque  $J_R$ , which is assumed to be proportional to the angular velocity (viscous friction);  $J_R = -\rho \dot{\varphi}_e$ , where  $\rho$  is the coefficient of friction.

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.



**Figure 3.7** A mechanical analogy illustrating the linear and nonlinear performance of the PLL. (Symbols defined in text.)

- The torque  $J_A$  generated by the gravity of the weight  $G$ ;  $J_A = rG$ , where  $r$  is the radius of the cylinder.

Introducing these individual torques into [Eq. \(3.45\)](#) yields

$$\ddot{\varphi}_e + \frac{\rho}{T} \dot{\varphi}_e + \frac{Mag}{T} \sin \varphi_e = \frac{r}{T} G(t) \quad (3.46)$$

As in the case of the PLL, we can write this equation in a normalized form. If we introduce the substitutions

$$(3.47)$$

$$\omega'_n = \left( \frac{Mag}{T} \right)^{1/2}$$
$$\zeta' = \frac{\rho}{2\sqrt{MagT}}$$

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

[Equation \(3.46\)](#) is converted into

$$\ddot{\varphi}_e + 2\zeta'\omega'_n\dot{\varphi}_e + \omega'^2_n \sin \varphi_e = \frac{r}{T}G(t) \cong G(t) \quad (3.48)$$

This nonlinear differential equation for the deflection angle  $\varphi_e$  looks very much like the nonlinear differential equation of the PLL according to [Eq. \(3.44\)](#). A slight difference in the second term exists, however. In the case of the PLL, the second term contains the factor  $\cos \theta_e$ , whereas for the pendulum the coefficient of the second term is the constant  $2\zeta' \omega'_n'$ . Strictly speaking, the pendulum of [Fig. 3.7](#) would only be an accurate analogy of the PLL if the friction varied with the cosine of the deflection angle. This would be true if the damping factor  $\zeta'$  were not a constant but instead varied with  $\cos \varphi_e$ . As a consequence, the momentary friction torque would be positive for

$$-\frac{\pi}{2} < \varphi_e < \frac{\pi}{2}$$

that is, when the position of the pendulum is in the lower half of a circle around the cylinder shaft. On the other hand, the momentary friction torque would be negative for

$$\frac{\pi}{2} < \varphi_e < \frac{3\pi}{2}$$

that is, when the position of the pendulum is in the upper half of this circle. A negative friction is hard to imagine, of course, but let us assume for the moment that  $\zeta' \cong \cos \varphi_e$  is valid. Imagine further that the weight  $G$  is large enough to make the pendulum tip over and continue to rotate forever around its axis (provided the rope is long enough). Because of the nonconstant torque generated by the mass  $M$  of the pendulum, this oscillation will be nonharmonic. During the time when the pendulum swings through the lower half of the circle ( $-\pi/2 < \varphi_e < \pi/2$ ), its average angular velocity is greater than its velocity during the time when it swings through the upper half ( $\pi/2 < \varphi_e < 3\pi/2$ ). The positive friction torque averaged over the lower semicircle is therefore greater in magnitude than the negative friction torque averaged over the upper semicircle. This means that the friction torque averaged over one full revolution stays positive; hence, it is acceptable to state that the coefficient of friction  $\zeta'$  varies with the cosine of  $\varphi_e$ . The mathematical pendulum is therefore a reasonable approximation for the PLL.

Comparing the PLL with this mathematical pendulum, we find the following analogies:

- The phase error  $\theta_e$  of the PLL corresponds to the angle of deflection  $\varphi_e$  of the pendulum.
- The natural frequency  $\omega_n$  of the PLL corresponds to the natural (or resonant) frequency of the pendulum  $\omega'_n$ .

- The damping factor  $\zeta$  of the PLL corresponds to the damping factor  $\zeta'$  of the pendulum, which results from viscous friction.
- The weight  $G$  on the platform corresponds to a reference phase disturbance according to the relation

$$\ddot{\theta}_1 + \dot{\theta}_1 \frac{\omega_n^2}{K_0 K_d / N} \equiv G(t) \quad (3.49)$$

Let's now determine what the physical meaning of the term  $\ddot{\theta}_1 + \dot{\theta}_1 \frac{\omega_n^2}{K_0 K_d / N}$  is in Eq. (3.49). We first assume that the frequency of the reference signal has an arbitrary value:

$$\omega_1 = \omega_0' + \Delta\omega(t)$$

where  $\Delta\omega(t)$  can be considered the frequency offset of the reference signal. The reference signal  $u_1(t)$  can therefore be written in the form

$$u_1(t) = U_{10} \sin \left\{ \int_0^t [\omega_0' + \Delta\omega(t)] dt = U_{10} \sin [\omega_0' t + \theta_1(t)] \right\}$$

Consequently, the phase  $\theta_1(t)$  is given by

$$\theta_1(t) = \int_0^t \Delta\omega(t) dt$$

From this, it becomes immediately evident that the first derivative represents the momentary frequency offset:

$$\dot{\theta}_1(t) = \Delta\omega(t)$$

whereas the second derivative signifies the rate of change of the frequency offset:

$$\ddot{\theta}_1(t) = \frac{d}{dt} \Delta\omega(t) = \Delta\dot{\omega}$$

The weight  $G$  placed on the platform is thus equivalent to a weighted sum of the frequency offset  $\Delta\omega(t)$  and its rate of change  $\Delta\dot{\omega}(t)$

$$G(t) \equiv \Delta\dot{\omega} + \frac{\omega_n^2}{K_0 K_d / N} \Delta\omega \quad (3.50)$$

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

This simple correspondence paves the way toward understanding the quite complex dynamic performance of a PLL in the locked and unlocked states. To see what happens to a PLL when phase and/or frequency steps of arbitrary size are applied to its reference input, we must place the corresponding weight  $G(t)$  given by Eq. (3.50) on the platform and observe the response of the pendulum. The notation  $G(t)$  should emphasize that  $G$  must not necessarily be a constant, but can also be a function of time, as would be the case when an impulse is applied.

Let's first consider the trivial case of no weight on the platform. The pendulum is then at rest in a vertical position,  $\varphi_e = 0$ . This corresponds to the PLL operating at its center frequency  $\omega_0'$  with zero frequency offset ( $\Delta\omega = 0$ ) and zero phase error ( $\theta_e = 0$ ).

What happens if the frequency of the reference signal is changed *slowly*? The rate of change of the reference frequency is assumed to be so low that the derivative term  $\Delta\dot{\omega}$  in Eq. (3.50) is negligible. A slow variation of the reference frequency corresponds to a slow increase of weight  $G$ , achieved by very carefully pouring a fine powder onto the platform. The analogy is given in this case by

$$G(t) \cong \frac{\omega_n^2}{K_0 K_d / N} \Delta\omega$$

The pendulum now starts to deflect, indicating that a finite phase error is established within the PLL. For small offsets of the reference frequency, the phase error  $\theta_e$  will be proportional to  $\Delta\omega$ . If the frequency offset reaches a critical value, called *the hold range*, the deflection of the pendulum is just  $90^\circ$ . This is the static limit of stability. With the slightest disturbance, the pendulum would now tip over and rotate around its axis forever. This corresponds to the case where the PLL is no longer able to maintain phase tracking and consequently unlocks. One full revolution of the pendulum equals a phase error of  $2\pi$ . Because the pendulum is now rotating permanently, the phase error increases toward infinity.

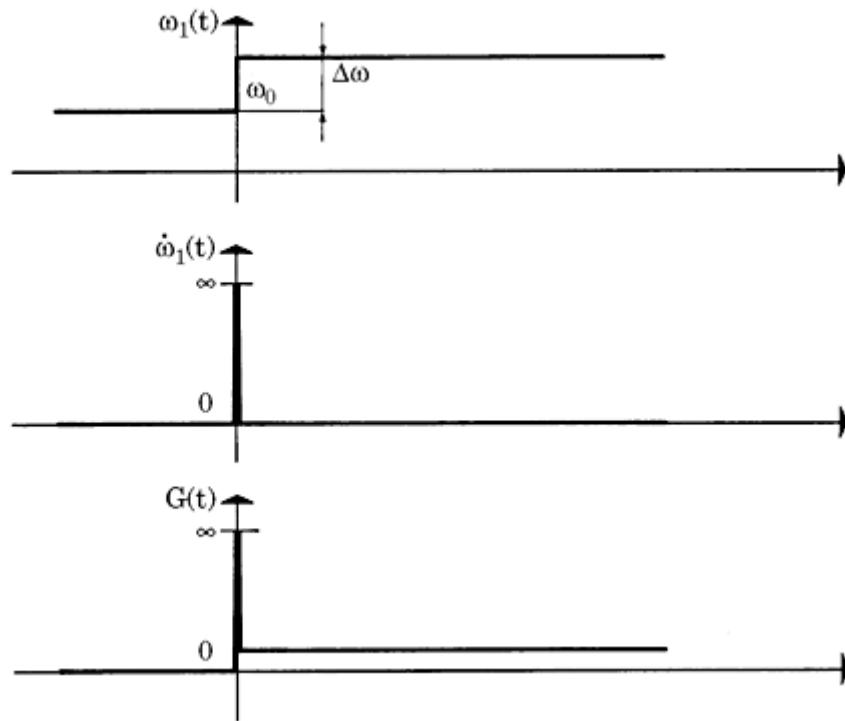
Another interesting case is given by a step change of the reference frequency at the input of the PLL. When a frequency step of the size  $\Delta\omega$  is applied at  $t = 0$ , the angular frequency of the reference signal is

$$\omega_1(t) = \omega_0' + \Delta\omega \cdot u(t)$$

where  $u(t)$  is the unit-step function. The first derivative  $\Delta\dot{\omega}(t)$  therefore shows a delta function at  $t = 0$ ; this is written as

$$\Delta\dot{\omega}(t) = \Delta\omega \delta(t)$$

and is plotted in Fig. 3.8. What will now be the weight  $G(t)$  required to simulate this condition? As also shown in Fig. 3.8, the weight function should be a superposition of a step function and a delta (impulse) function. In practice, this can be simulated by dropping an appropriate weight from some height onto the platform. The impulse is generated when the weight hits the platform. To get



**Figure 3.8** The weight function  $G(t)$  required to simulate a frequency step applied to the reference input of the PLL. ( $\omega_1$  = angular frequency of the reference signal;  $\Delta\omega$  = frequency step applied at  $t = 0$ .)

a narrow and steep impulse, the stroke should be *elastic*. If this is done, the pendulum will show a transient response, mostly in the form of damped oscillation. If a relatively small weight is *dropped* onto the platform, the final deflection of the pendulum will be the same as if the weight had been placed *smoothly* onto the platform. If the pendulum is not heavily overdamped, however, its peak deflection  $\hat{\varphi}_e$  will be considerably greater than its final deflection. If we increase the weight dropped onto the platform, we will observe a situation where the peak deflection exceeds  $90^\circ$ , but not  $180^\circ$ , and the final deflection is less than  $90^\circ$ . We thus conclude that a linear PLL can operate stably when the phase error  $\theta_e$  momentarily exceeds the value of  $90^\circ$ . If the weight dropped onto the platform is increased ever further, the peak deflection will exceed  $180^\circ$ . The pendulum now tips over and performs a number of revolutions around its axis, but it will probably come to rest again after some time.

The weight that caused the system to unlock (at least temporarily) is observed to be considerably smaller than the weight that represented the hold range. We therefore must define another critical frequency offset—in other words, the offset that causes the PLL to unlock when it is applied as a step. This frequency step is called the *pull-out range*.

Keep in mind that the *pull-out range* of a PLL is markedly smaller than its *hold range*. The pull-out range may be considered the dynamic limit of stability. The PLL always stays locked as long as the frequency steps applied to the system do not exceed the pull-out range.

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

A third way can be employed to cause a PLL to unlock from an initially stable operating point. If the frequency of the reference signal is increased linearly with time, we have

$$\Delta\omega(t) = \Delta\dot{\omega} \cdot t$$

where  $\Delta\dot{\omega}$  is the rate of change of the angular frequency. In the mechanical analogy, this corresponds to a weight  $G$  that also builds up linearly with time. This can be realized by feeding a material onto the platform at an appropriate feed rate. It becomes evident that too rapid a feed rate acts on the pendulum like the impulse generated in the last example by dropping a weight onto the platform. As has been shown in [Sec. 3.4](#), the rate of change of the reference frequency must always be smaller than  $\omega_n^2$  to keep the system locked:

$$\Delta\dot{\omega} < \omega_n^2$$

These examples have demonstrated that three conditions are necessary if a PLL system is to maintain phase tracking:

- The angular frequency of the reference signal must be within the *hold range*.
- The maximum frequency step applied to the reference input of a PLL must be smaller than the *pull-out range*.
- The rate of change of the reference frequency must be lower than  $\omega_n^2$ .

Whenever a PLL has lost tracking because one of these conditions has not been fulfilled, the question arises as to whether it will return to stable operation when all the necessary conditions are met again. The answer is clearly *no*. When the reference frequency exceeded the hold range, the pendulum in our analogy tipped over and started rotating around its axis. If the weight on the platform were reduced to a value slightly less than the critical limit that caused instability, the pendulum would nevertheless continue to rotate because there is enough kinetic energy stored in the mass to maintain the oscillation. If there were no friction at all, the pendulum would continue to rotate even if the weight  $G$  were reduced to zero. Fortunately friction exists, so the pendulum will decelerate if the weight is decreased to an appropriate level. For a PLL, this means that build-up of the phase error will decelerate if the reference-frequency offset is decreased below another critical value, the *pull-in frequency*. If the slope of the average phase error becomes smaller, the (down-scaled) frequency  $\omega_2'$  of the VCO more and more approaches the frequency of the reference signal, and the system will finally lock. The *pull-in frequency*  $\Delta\omega_P$  is markedly smaller than the *hold range*, as can be expected from the mechanical analogy. The pull-in process is a relatively slow one, as will be demonstrated in [Sec. 3.9.3](#).

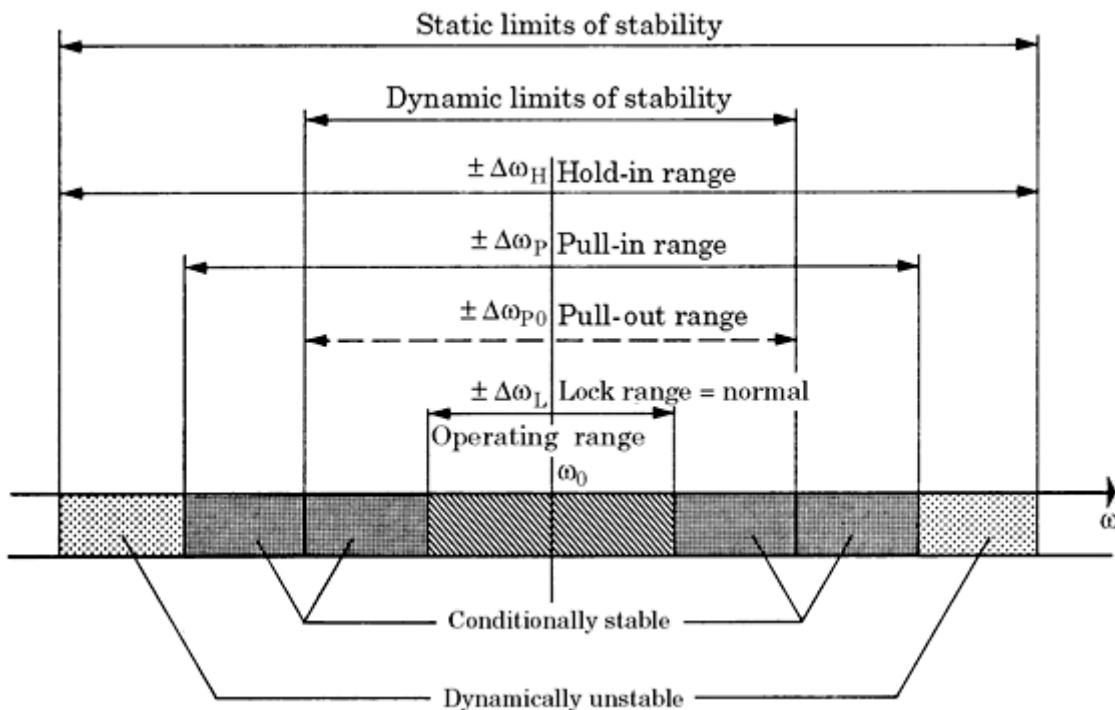
In most practical applications, it is desired that the locked state be obtained within a short time period. Suppose again that the weight put on the platform of the mechanical model is large enough to cause sustained rotation of the pendulum. It is easily shown that the pendulum can be brought to rest within one single revolution if the weight is suddenly decreased below another critical

value. This implies that a PLL can become locked within one single-beat note between reference frequency and output frequency, provided the frequency offset  $\Delta\omega$  is reduced below a critical value called the *lock range*. This latter process is called the *lock-in process*. The lock-in process is much faster than the pull-in process, but the *lock range* is smaller than the *pull-in range*.

The mechanical model has shown there are four key parameters specifying the frequency range in which the PLL can be operated. [Figure 3.9](#) is a graphical representation of these parameters. The four key parameters can be summarized as follows:

- *The hold range*  $\Delta\omega_H$ . This is the frequency range in which a PLL can statically maintain phase tracking. A PLL is conditionally stable only within this range.
- *The pull-out range*  $\Delta\omega_{PO}$ . This is the dynamic limit for stable operation of a PLL. If tracking is lost within this range, a PLL normally will lock again, but this process can be slow if it is a pull-in process.
- *The pull-in range*  $\Delta\omega_P$ . This is the range within which a PLL will always become locked, but the process can be rather slow.
- *The lock range*  $\Delta\omega_L$ . This is the frequency range within which a PLL locks in one single-beat note between reference frequency and output frequency. Normally, the operating-frequency range of a PLL is restricted to the lock range.

The discussion of the mechanical analogy did not provide numerical solutions for these four key parameters. We will derive such expressions in [Sec. 3.9](#). The



**Figure 3.9** Scope of the static and dynamic limits of stability of a linear second-order PLL.

**Any use is subject to the Terms of Use as given at the website.**

quantitative relationships among these four parameters are plotted in [Fig. 3.9](#) for most practical cases. We can state in advance that the hold range  $\Delta\omega_H$  is greater than the three remaining parameters. Furthermore, we know that the pull-in range  $\Delta\omega_P$  must be greater than the lock range  $\Delta\omega_L$ . The pull-in range  $\Delta\omega_P$  is greater than the pull-out range  $\Delta\omega_{PO}$  in most practical designs, so we get the simple inequality

$$\Delta\omega_L < \Delta\omega_{PO} < \Delta\omega_P < \Delta\omega_H$$

In many texts, the term *capture range* can also be found. In most cases, capture range is an alternative expression for *lock range*; sometimes, it is also used to mean *pull-in range*. The differentiation between lock-in and pull-in ranges is not clearly established in some books, but we will maintain it throughout this text.<sup>1</sup>

## Key Parameters of the PLL

In [Sec. 3.8](#), it was demonstrated that the dynamic performance of the PLL is governed by a set of key parameters:

- The lock range  $\Delta\omega_L$
- The pull-out range  $\Delta\omega_{PO}$
- The pull-in range  $\Delta\omega_P$
- The hold range  $\Delta\omega_H$

Here, we will define parameters relating to the time required for the PLL to get locked:

- *The lock time*  $T_L$ . The time the PLL needs to get locked when the acquisition process is a (fast) lock-in process
- *The pull-in time*  $T_P$ . This is the time the PLL needs to get locked when the acquisition process is a (slow) pull-in process

To design a PLL system, we need equations that tell us how these key parameters depend on the parameters of the circuit—that is, the time constants  $\tau_1$  and  $\tau_2$  of the loop filter, the gain factors  $K_d$  or  $K_P$ ,  $K_0$ , and  $K_a$  (when the active lead-lag filter is used), and the divider ratio  $N$  of an optional down scaler. In the following, we will derive a number of approximate design equations for these parameters. Unfortunately, these equations depend on the type of phase detectors and loop filters used. To make the design easier, the results will be shown in tables. Moreover, we will demonstrate in [Chap. 10](#) that the design can be performed by a dedicated computer program developed by the author.

## Hold range $\Delta\omega_H$

First of all, let us state that the hold range is a parameter of more academic interest. The hold range is the frequency range in which a PLL is able to maintain

**Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.**

lock statically. As we have seen in [Sec. 3.8](#), the PLL locks out forever when the frequency of the input signal exceeds the hold range, so most practitioners don't even worry about the actual value of this parameter.

The magnitude of the hold range is obtained by calculating that frequency where the phase error is at its maximum. As we have seen in [Sec. 2.4](#), this maximum value depends on the type of phase detector used. With a multiplier phase detector, the PLL can maintain lock at a phase error slightly less than  $90^\circ$ . The same holds true for the EXOR phase detector. When the JK-flipflop phase detector is used, however, the PLL can stay locked up to a phase error of  $180^\circ$ , and for the PFD this value is even  $360^\circ$ . The equation for the hold range  $\Delta\omega_H$  will therefore be different for each type of phase detector. Let us first analyze the hold range for the multiplier phase detector.

**Phase detector type 1.** To get the hold range  $\Delta\omega_H$ , we must determine the frequency for which the steady state phase error becomes  $90^\circ$ . At the limit of the hold range, the input frequency  $\omega_1$  is given by

$$\omega_1 = \omega_0' + \Delta\omega_H$$

For the phase signal  $\theta_1(t)$ , we therefore get

$$\theta_1(t) = \Delta\omega_H \cdot t$$

The Laplace transform of the phase signal then becomes

$$\Theta_1(s) = \frac{\Delta\omega_H}{s^2}$$

The phase error can now be calculated according to [Eq. \(3.27\)](#)

$$\Theta_e(s) = \Theta_1(s)H_e(s) = \frac{\Delta\omega_H}{s^2} \frac{s}{s + K_0K_dF(s)/N}$$

Using the final-value theorem of the Laplace transform, we calculate the final phase error  $\theta_e(\infty)$  in the time domain

$$\theta_e(\infty) = \lim_{s \rightarrow 0} s\Theta_e(s) = \frac{\Delta\omega_H}{K_0K_dF(0)/N} \quad (3.51)$$

Remember that the PLL network was linearized when the Laplace transform was introduced. Consequently, [Eq. \(3.51\)](#) is valid for small values of  $\theta_e$  only. For greater values of phase error, we would have to write

$$\sin \theta_e(\infty) = \lim_{s \rightarrow 0} s\Theta_e(s) = \frac{\Delta\omega_H}{K_0K_dF(0)/N}$$

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

At the limit of the hold range,  $\theta_e = \pi/2$  and  $\sin \theta_e = 1$ . Therefore, we obtain for the hold range the expression

$$\Delta\omega_H = K_0 K_d F(0)/N$$

The DC gain  $F(0)$  of the loop filter depends on the filter type. For the passive lead-lag filter, the DC gain  $F(0) = 1$ . For the active lag filter, the DC gain is  $F(0) = K_a$ . For the active PI filter,  $F(0) = \infty$ , at least theoretically. When phase detector type 1 is used, we get for the hold range

$$\Delta\omega_H = \begin{cases} K_0 K_d / N & \text{(passive lead-lag)} \\ K_0 K_d K_a / N & \text{(active lead-lag)} \\ \infty & \text{(active PI)} \end{cases} \quad (3.52)$$

When the active PI filter is used, the actual hold range is limited by the frequency range covered by the VCO.

**Phase detector type 2.** When the EXOR phase detector is chosen, the maximum phase error in the steady state can be  $\pi/2$ . In contrast to phase detector type 1, the equation  $\bar{u}_d = K_d \theta_e$  is valid for all values of  $\theta_e$  in the range  $-\pi/2 < \theta_e < \pi/2$ . Hence, we get from Eq. (3.51)

$$\Delta\omega_H = \frac{K_0 K_d F(0) \pi/2}{N} \quad (3.53)$$

Note again that  $F(0)$ —the loop filter gain at DC—depends on the type of loop filter chosen, as described earlier.

**Phase detector type 3.** When the JK-flipflop phase detector is chosen, the average phase detector output  $\bar{u}_d$  is proportional to the phase error  $\theta_e$  over the whole range  $-\pi < \theta_e < \pi$ . By an analogous argumentation, we get for the hold range

$$\Delta\omega_H = \frac{K_0 K_d F(0) \pi}{N} \quad (3.54)$$

**Phase detector type 4a.** When the PFD with voltage output is used, the average phase detector output  $\bar{u}_d$  is proportional to the phase error over the range  $-2\pi < \theta_e < 2\pi$ . Hence, we get for the hold range

$$\Delta\omega_H = \frac{K_0 K_d F(0) 2\pi}{N} \quad (3.55)$$

To get the hold range for the three different types of loop filters, we have to insert the correct DC gain into  $F(0)$  in Eq. (3.55). For the previously discussed types of phase detector,

we put  $F(0) = 1$  for the passive lead-lag filter,  $F(0) = K_a$  for the active lead-lag filter, and  $F(0) = \infty$  for the active PI filter. As we shall now see, we will have to set  $F(0) = \infty$  for all filter types. What is the reason for this?

Let us consider a cascade connection of a voltage output PFD (cf. [Fig. 2.11](#)) and a passive lead-lag filter (cf. [Fig. 2.17a](#)). When the PFD output floats, the

charge on capacitor remains unchanged—that is, the voltage on the capacitor stays constant. (In practice, the capacitor would be discharged by leakage currents, of course.) This implies that the passive lead-lag filter behaves like a real integrator when driven by a tri-state signal! In other words, the passive lead-lag filter has infinite gain at  $f = 0$  under this condition. Volgers has shown<sup>15</sup> that the transfer function of the passive lead-lag filter must be modified when it is driven by the PFD. It shows up that the transfer function is approximately given by

$$F(s) \approx \frac{1 + s\tau_2}{s(\tau_1 + \tau_2)} \quad (3.56)$$

When the active lead-lag filter is used, the transfer function is approximated by

$$F(s) \approx K_a \frac{1 + s\tau_2}{s\tau_1} \quad (3.57)$$

where  $K_a = C_1/C_2$  (cf. Fig. 2.19a). When the active PI filter is chosen, however, it does not matter whether the filter is driven by a “normal” signal source or by a device having a tri-state output. In any case, the PI filter acts as an integrator whose transfer function is given by

$$F(s) \approx \frac{1 + s\tau_2}{s\tau_1} \quad (3.58)$$

Consequently, the DC gain  $F(0)$  for all three filter types is infinite when driven by a tri-state source. Hence, the hold range for phase detector 4a becomes

$$\Delta\omega_H = \infty \quad (3.59)$$

Note: Inversion of polarity is discarded in Eqs. (3.57) and (3.58).

**Phase detector type 4b.** For the PFD with current output, no output current flows when both current sources are off (cf. Fig. 2.17). Therefore, the PFD with current output behaves like a PFD with tri-state output as discussed earlier. Also, here the hold range becomes infinite

$$\Delta\omega_H = \infty \quad (3.60)$$

## Lock range $\Delta\omega_L$ and lock time $T_L$

As in the previous section, here we analyze the lock range and the lock time separately for each type of phase detector.

**Phase detector type 1.** The magnitude of the lock range can be obtained by a simple consideration. We assume that the PLL initially is not locked and that the reference frequency is  $\omega_1 = \omega_0' + \Delta\omega$ . The reference signal of the PLL is then given by

$$u_1(t) = U_{10} \sin(\omega_0' t + \Delta\omega \cdot t)$$

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

and the output signal by

$$u_2(t) = U_{20} \operatorname{rect}(\omega_0' t)$$

where “rect” denotes a symmetrical square wave signal.

Using Eq. (2.16), the phase detector will deliver an output signal given by

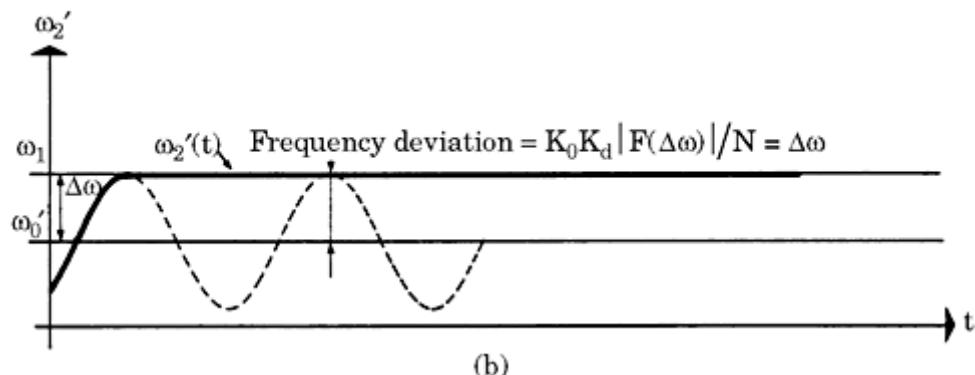
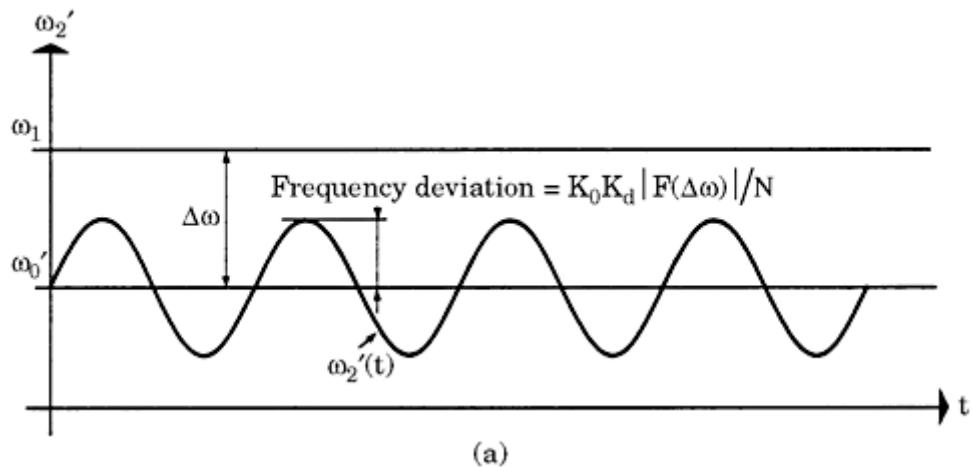
$$u_d(t) = K_d \sin(\Delta\omega \cdot t) + \text{higher-frequency terms}$$

The higher-frequency terms can be discarded because they will be almost entirely filtered out by the loop filter. At the output of the loop filter, there appears a signal  $u_f(t)$  given by

$$u_f(t) = K_d |F(\Delta\omega)| \sin(\Delta\omega \cdot t)$$

This is an AC signal that causes a frequency modulation of the VCO.

The peak frequency deviation is equal to  $K_d K_0 |F(\Delta\omega)|$ . When a down scaler is used (cf. Fig. 2.1), this frequency deviation is scaled down by factor  $N$ . The peak frequency deviation appearing at the lower input of the phase detector (cf. Fig. 2.1) is therefore given by  $K_d K_0 |F(\Delta\omega)| / N$ . In Fig. 3.10, the frequency  $\omega_2'(t)$  of the VCO



**Figure 3.10** Lock-in process. (a) The peak frequency deviation is smaller than the offset  $\Delta\omega$ ;

therefore, a fast lock-in process cannot take place. (b) The peak frequency deviation is exactly as large as the actual frequency offset; thus, the PLL will become locked after a very short time.

is plotted against time for two cases. In [Fig. 3.10a](#), the peak frequency deviation is less than the offset  $\Delta\omega$  between the reference frequency  $\omega_1$  and the down-scaled frequency  $\omega_0'$ . Hence, a lock-in process will not take place, at least not instantaneously.

[Figure 3.10b](#) shows a special case, however, where the peak frequency deviation is just as large as the frequency offset  $\Delta\omega$ . The frequency  $\omega_2'$  of the VCO output signal develops as shown by the solid line. When the frequency deviation is at its largest,  $\omega_2'$  exactly meets the value of the reference frequency  $\omega_1$ . Consequently, the PLL locks within one single-beat note between the reference and output frequencies. This corresponds exactly to the lock-in process described in [Sec. 3.8](#) by means of the mechanical analogy. Under this condition, the difference  $\Delta\omega$  is identical with the lock range  $\Delta\omega_L$ , and we have

$$\frac{K_0 K_d}{N} |F(\Delta\omega_L)| = \Delta\omega_L$$

This is a nonlinear equation for  $\Delta\omega_L$ . Its solution becomes very simple, however, if an approximation is introduced for  $|F(\Delta\omega_L)|$ . It follows from practical considerations that the lock range is always much greater than the corner frequencies  $1/\tau_1$  and  $1/\tau_2$  of the loop filter. For the filter gain  $|F(\Delta\omega_L)|$ , we can therefore make the following approximations:

$$|F(\Delta\omega_L)| \approx \begin{cases} \frac{\tau_2}{\tau_1 + \tau_2} & \text{for the passive lead-lag filter} \\ K_a \frac{\tau_2}{\tau_1} & \text{for the active lead-lag filter} \\ \frac{\tau_2}{\tau_1} & \text{for the active PI filter} \end{cases}$$

Moreover,  $\tau_2$  is normally much smaller than  $\tau_1$ , so we can use the simplified relationship

$$|F(\Delta\omega_L)| \approx \begin{cases} \tau_2/\tau_1 & \text{for the passive lag filter} \\ K_a \tau_2/\tau_1 & \text{for the active lag filter} \\ \tau_2/\tau_1 & \text{for the active PI filter} \end{cases}$$

Making use of the substitutions [Eqs. [\(3.13\)](#) to [\(3.15\)](#)] and assuming high-gain loops, we get

$$\Delta\omega_L \approx 2\zeta\omega_n \quad (3.61)$$

for all types of loop filters.

Knowing the approximate size of the lock range, we are certainly interested in having some indication of the lock-in time. When the PLL locks in quickly, the signals  $u_d$  and  $u_f$  perform (for  $\zeta < 1$ ) a damped oscillation, whose angular frequency is approximately  $\omega_n$ . As [Fig. 3.6](#)

shows, the transients die out in

about one cycle of this oscillation; hence, it is reasonable to state that the lock-in time is

$$T_L \approx \frac{2\pi}{\omega_n} \quad (3.62)$$

which is valid for any type of loop filter.  $T_L$  is also referred to as settling time.

**Phase detector type 2.** When the EXOR phase detector is chosen, the lock range can be determined by considerations analogous to those made for the multiplier phase detector. We assume the PLL is initially out of lock and that both signals  $u_1$  and  $u_2'$  are symmetrical square waves. The reference frequency  $\omega_1$  is offset from the center frequency  $\omega_0'$  by  $\Delta\omega$ . Then for  $u_1$  and  $u_2'$  we have

$$\begin{aligned} u_1(t) &= U_{10} \operatorname{rect}(\omega_0't + \Delta\omega \cdot t) \\ u_2(t) &= U_{20} \operatorname{rect}(\omega_0't) \end{aligned}$$

where  $U_{10}$  and  $U_{20}$  are the amplitudes of the square-wave signals. The phases  $\theta_1$  and  $\theta_2'$  are then given by

$$\begin{aligned} \theta_1(t) &= \omega_0't + \Delta\omega \cdot t \\ \theta_2(t) &= \omega_0't \end{aligned}$$

The phase error  $\theta_e$  therefore is

$$\theta_e = \Delta\omega \cdot t$$

which is a ramp function. Checking Fig. 2.7 again, we note the average phase detector output signal  $\overline{u_d}$  represents a triangular signal when the phase error ramps up linearly with time, its peak amplitude being  $K_d \pi/2$ . Therefore,  $\overline{u_d}$  can be written as

$$\overline{u_d}(t) = K_d \frac{\pi}{2} \operatorname{tri}(\Delta\omega \cdot t)$$

where “tri” stands for “triangular waveform.” This signal is depicted in the upper trace of Fig. 3.11. The average output signal of the loop filter is now given by

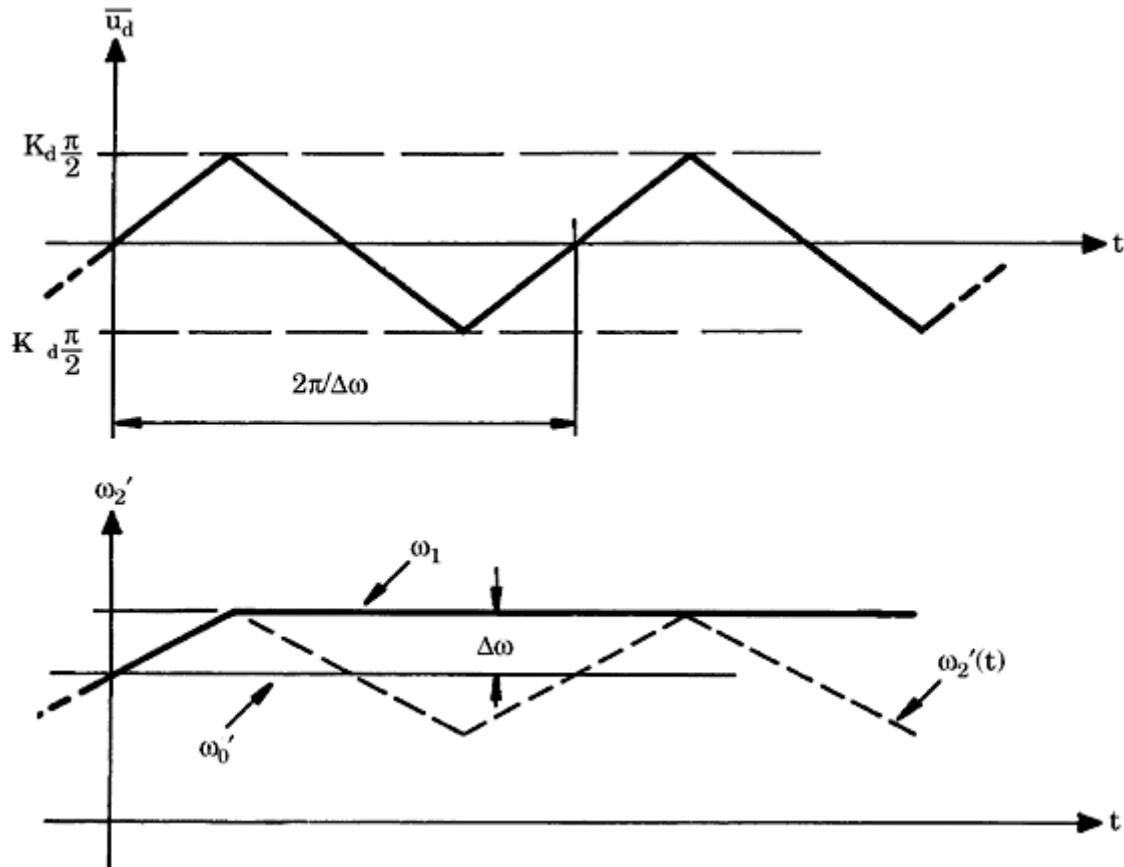
$$u_f(t) = F(\Delta\omega) K_d \frac{\pi}{2} \operatorname{tri}(\Delta\omega \cdot t)$$

This signal is shown in the lower trace of Fig. 3.11. This signal modulates the frequency of the VCO such that its peak deviation becomes

$$\Delta\hat{\omega}_2 = F(\Delta\omega) K_0 K_d \frac{\pi}{2} \quad (3.63)$$

When the division ratio of the (optional) down scaler is  $N$ , the peak frequency deviation at

the lower input of the phase detector (cf. Fig. 2.1) becomes  $\Delta\hat{\omega}_2/N$ .



**Figure 3.11** Waveforms of the average phase detector output signal  $\bar{u}_d(t)$  (upper trace) and the scaled-down radian frequency  $\omega_2'$  (lower trace) for the case where the radian frequency offset  $\Delta\omega$  is identical with the lock range  $\Delta\omega_L$ .

It is easily seen now that the PLL acquires lock within one beat note when  $\Delta\hat{\omega}_2/N$  equals the radian frequency offset  $\Delta\omega$

$$\Delta\hat{\omega}_2/N = \Delta\omega \quad (3.64)$$

Under this condition,  $\Delta\omega$  is identical with the lock range  $\Delta\omega_L$ . Combining Eqs. (3.63) and (3.64), we obtain the nonlinear equation.

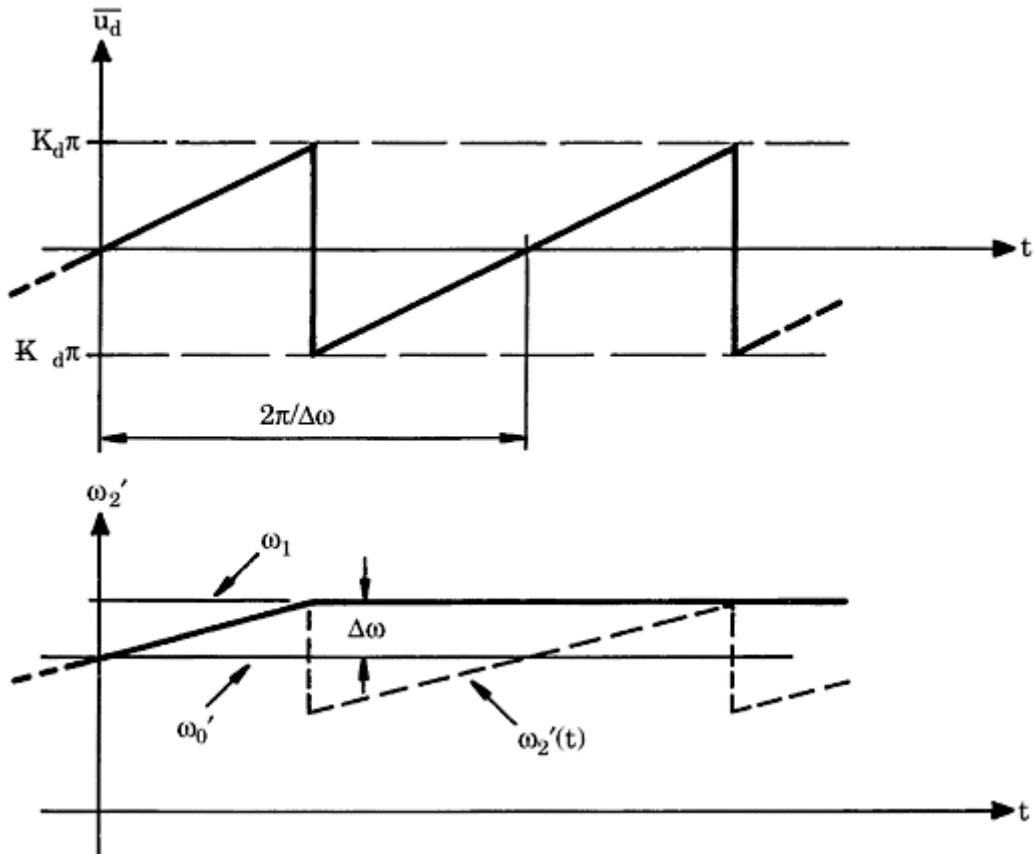
$$F(\Delta\omega_L)K_0K_d\frac{\pi}{2}/N = \Delta\omega_L$$

Using the same substitutions as for the type 1 phase detector, we get the simplified expression

$$\Delta\omega_L \approx \pi\zeta\omega_n \quad (3.65)$$

[Equation \(3.65\)](#) is valid for any type of loop filter. For the lock time  $T_L$ , we obtain the same approximation made for type 1 phase detector [[Eq. \(3.62\)](#)].

When we compare the lock range obtained for the EXOR phase detector with the lock range obtained for the multiplier phase detector, we observe that  $\Delta\omega_L$  is larger by a factor of  $\pi/2$  in case of the EXOR.



**Figure 3.12** Waveforms of the average phase detector output signal  $\bar{u}_d(t)$  (upper trace) and the scaled-down radian frequency  $\omega_2'$  (lower trace) for the case where the radian frequency offset  $\Delta\omega$  is identical with the lock range  $\Delta\omega_L$ .

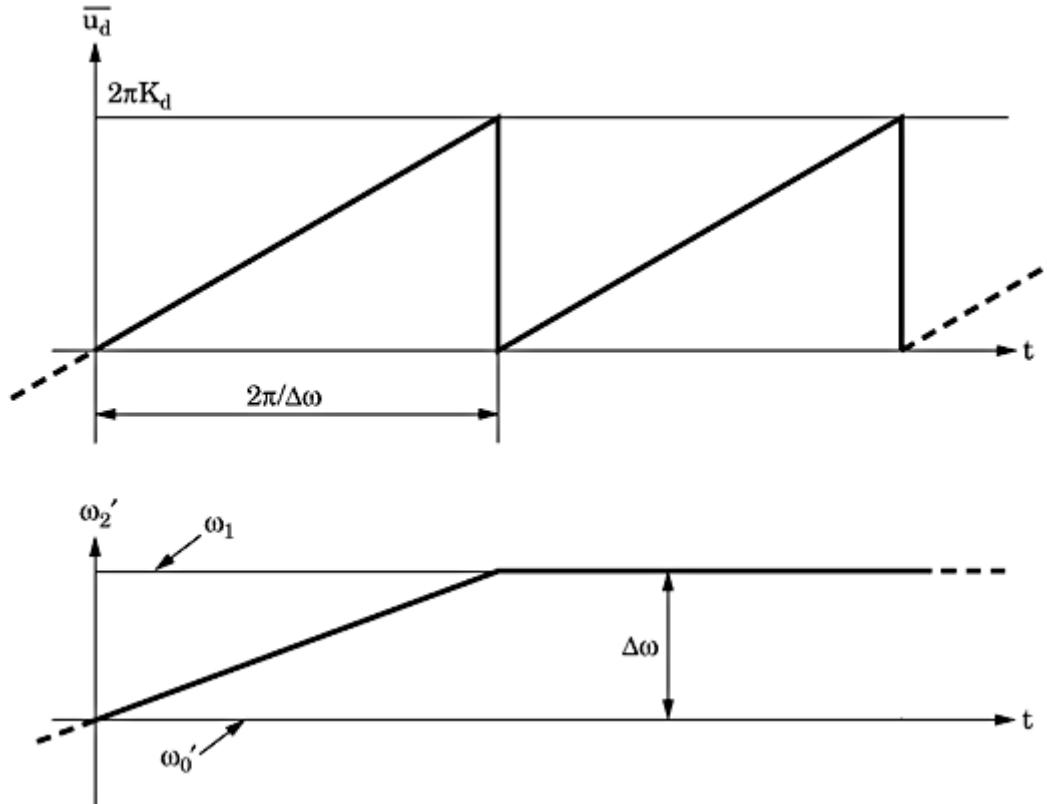
**Phase detector type 3.** The lock range of the PLL using a type 3 phase detector can be computed by employing a similar consideration as that made for type 2 (EXOR). If we assume again that the PLL is initially out of lock and that the offset between reference frequency  $\omega_1$  and center frequency  $\omega_0'$  is again  $\Delta\omega$ , the phase error becomes a ramp function again. Because the average output signal of the JK-flipflop  $\bar{u}_d$  varies in a sawtooth-like fashion with phase error (Fig. 2.10), the average signal  $\bar{u}_d$  will also be a sawtooth function, as shown in the upper curve of Fig. 3.12.

Now the frequency of the VCO is modulated in a sawtooth-like manner (see the lower curve of Fig. 3.12). If the frequency offset  $\Delta\omega$  is chosen such that the curve just touches the  $\omega_1$  line,  $\Delta\omega$  equals the lock range  $\Delta\omega_L$ . Using an analogous consideration, we get the approximation

$$\Delta\omega_L \approx 2\pi\zeta\omega_n \quad (3.66)$$

The lock range is larger by a factor of 2 than the lock range of a PLL using the EXOR phase detector. For the lock-in time  $T_L$ , Eq. (3.62) still holds true.

**Phase detector type 4a.** The waveforms for  $\overline{u_d}$  and  $\omega_2'$  versus time for the PFD are shown in Fig. 3.13. Again, the frequency of the VCO output signal is modulated in a sawtooth-like manner. Analogous to the preceding sections (EXOR



**Figure 3.13** Waveforms of the average phase detector output signal  $\bar{u}_d(t)$  (upper trace) and the scaled-down radian frequency  $\omega_2'$  (lower trace) for the case where the radian frequency offset  $\Delta\omega$  is identical with the lock range  $\Delta\omega_L$ . The PFD is used as a phase detector.

and JK-flipflop phase detector), we get the following approximation for the lock range:

$$\Delta\omega_L \approx 4\pi\zeta\omega_n \quad (3.67)$$

For the lock-in time approximation, Eq. (3.62) is still valid.

**Phase detector type 4b.** For the PFD with current output, the signals look the same as for the PFD with voltage output (cf. Fig. 3.13), but the upper waveform is now a current signal and has the peak value  $K_P 2\pi$ . An analysis similar to the preceding sections (EXOR, JK-flipflop, PFD with voltage output) yields a hold range given by

$$\Delta\omega_L \approx 4\pi\zeta\omega_n \quad (3.68)$$

where  $\zeta$  and  $\omega_n$  have been defined in Eq. (3.23). This expression is identical to that of the PFD with voltage output. For the lock time  $T_L$ , we get the same expression as for the other phase detectors [cf. Eq. (3.62)].

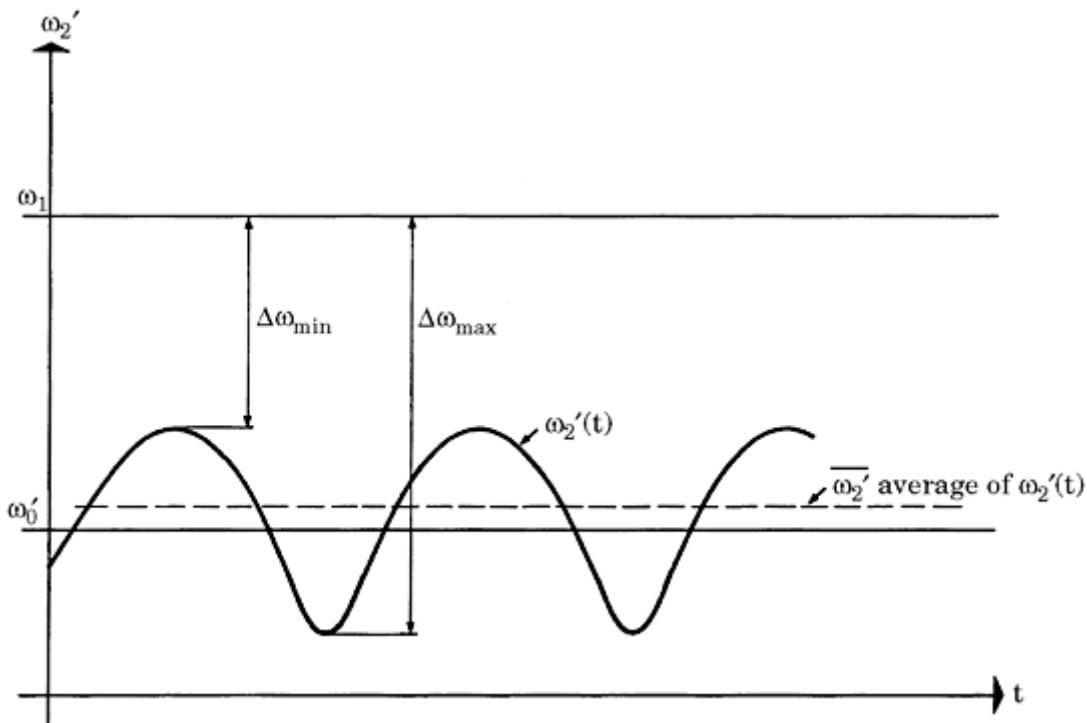
## Pull-in range $\Delta\omega_P$ and pull-in time $T_P$

**Phase detector type 1.** To determine the size of the pull-in range  $\Delta\omega_P$ , we assume that the PLL is not locked initially, that the frequency of the reference signal is  $\omega_1 = \omega_0' + \Delta\omega$ , and that the VCO initially operates at the center frequency

$\omega_0$ . Consequently, the average output signal  $\overline{u_d}$  of the phase detector is a sine wave having the frequency  $\Delta\omega$ , which is an AC signal. We now assume that the frequency offset  $\Delta\omega$  is so large that a lock-in process will not take place. Because the frequency offset  $\Delta\omega$  is generally much larger than the corner frequencies  $1/\tau_1$  and  $1/\tau_2$  of the loop filter, the loop filter will attenuate the  $\overline{u_d}$  signal. The loop filter output signal will also be a sine wave with radian frequency  $\Delta\omega$ , and the  $u_f$  signal will again modulate the frequency of the VCO output, as has been shown in Fig. 3.10.

If  $u_f$  were a perfect sine wave, its average value  $\overline{u_f}$  would be 0, hence the average output frequency of the VCO would stay stuck at its center frequency  $\omega_0$ . Under these conditions, the PLL would never pull in!

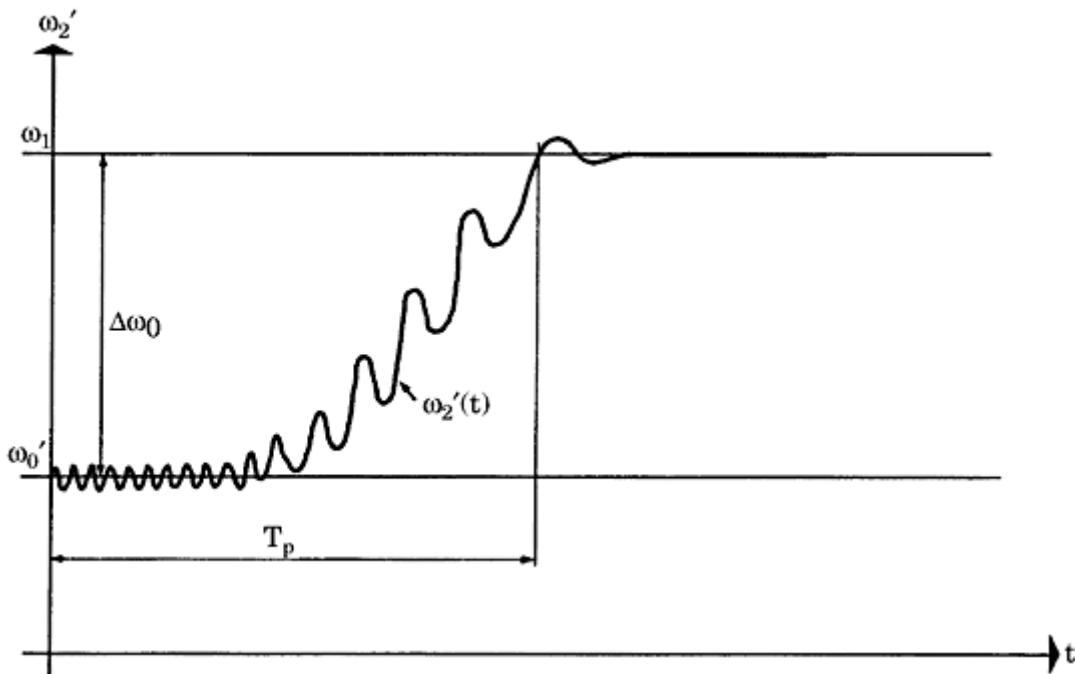
Fortunately, we have overlooked the fact that the difference  $\Delta\omega$  between reference frequency  $\omega_1$  and scaled down VCO output frequency  $\omega_2'(t)$  is not a constant; it is also varied by the frequency modulation of the VCO output signal. If the frequency  $\omega_2'(t)$  is modulated in the positive direction, the difference  $\Delta\omega$  becomes smaller and reaches some minimum value  $\Delta\omega_{\min}$ . If  $\omega_2'(t)$  is modulated in the negative direction, however,  $\Delta\omega$  becomes greater and reaches some maximum value  $\Delta\omega_{\max}$ . Because  $\Delta\omega(t)$  is not a constant, the VCO frequency is modulated nonharmonically—that is, the duration of the half-period in which  $\Delta\omega(t)$  is modulated in the positive direction becomes longer than that of the half-period in which  $\Delta\omega(t)$  is modulated in the negative sense. This is shown graphically in Fig. 3.14. As a consequence, the average frequency of the VCO  $\overline{\omega_2}$  is now



**Figure 3.14** In the unlocked state of the PLL, the frequency modulation of the VCO output signal is nonharmonic. This causes the average value of the VCO output frequency  $\overline{\omega_2}$  to be pulled in the direction of the reference frequency.

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.



**Figure 3.15** The pull-in process.

higher than it was without any modulation—in other words, the VCO frequency is pulled in the direction of the reference frequency. The asymmetry of the waveform  $\omega_2'(t)$  is greatly dependent on the value of the average offset  $\Delta\omega$ ; the asymmetry becomes more marked as  $\Delta\omega$  is decreased. If the average value of  $\omega_2'(t)$  is pulled somewhat in the direction of  $\omega_1$  (which is assumed to be greater than  $\overline{\omega_2'}$ ), the asymmetry of the  $\omega_2'(t)$  waveform becomes stronger. This in turn causes  $\overline{\omega_2'}$  to be pulled even more in the positive direction. This process is regenerative under certain conditions, so that the scaled-down output frequency  $\omega_2'$  finally reaches the reference frequency  $\omega_1$ . This phenomenon is called the *pull-in process* (Fig. 3.15). Mathematical analysis shows that a pull-in process occurs whenever the initial frequency offset  $\Delta\omega_0$  is smaller than a critical value, the pull-in range  $\Delta\omega_P$ . If, on the other hand, the initial frequency offset  $\Delta\omega_0$  is larger than  $\Delta\omega_P$ , a pull-in process does not take place because the pulling effect is not then regenerative.

The mathematical treatment of the pull-in process is quite cumbersome and is treated in more detail in [App. A](#). Here, we only give the final results. It is very important to note that the pull-in range depends on the type of loop filter.

The following are the formulas for the pull-in range:

- For the passive lead-lag filter

$$\begin{aligned}\Delta\omega_P &\approx \frac{4}{\pi} \sqrt{2\zeta\omega_n K_0 K_d / N - \omega_n^2} && \text{low-gain loops} \\ \Delta\omega_P &\approx \frac{4\sqrt{2}}{\pi} \sqrt{\zeta\omega_n K_0 K_d / N} && \text{high-gain loops}\end{aligned}\quad (3.69)$$

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

- For the active lead-lag filter

$$\begin{aligned}\Delta\omega_P &\approx \frac{4}{\pi} \sqrt{2\xi\omega_n K_0 K_d / N - \omega_n^2 / K_a} && \text{low-gain loops} \\ \Delta\omega_P &\approx \frac{4\sqrt{2}}{\pi} \sqrt{\xi\omega_n K_0 K_d / N} && \text{high-gain loops}\end{aligned}\quad (3.70)$$

- For the active PI filter

$$\Delta\omega_P \rightarrow \infty \quad (3.71)$$

Obviously, the PLL pulls in under any conditions when the loop filter is an active PI filter. As we know, the DC gain of this filter is (theoretically) infinite; hence, the slightest nonharmonicity of the  $u_d$  signal is sufficient to pull in the loop.

The pull-in process depicted in Fig. 3.15 can be easily explained by the mechanical analogy of Fig. 3.7. Initially, the frequency offset  $\Delta\omega_0$  is fairly large, and in the analogy the pendulum rotates at  $\Delta\omega_0/2\pi$  revolutions per second. The angular velocity decelerates slowly, however, and the pendulum comes to rest after some time. The “pumping” of the instantaneous frequency  $\omega_2'(t)$  is very characteristic of this process and is easily explained by the non-harmonic rotation of the pendulum caused by the gravity of its mass  $M$ . In fact, its angular velocity is greater at the lower “dead point” than at the higher one.

The duration of the pull-in process can also be computed from the mathematical analysis of the acquisition process (see App. A). The result also slightly depends on the type of loop filter used. The values given by the following formulas agree quite well with measurements on actual PLL circuits and with computer simulations.

They are only valid, however, if the initial frequency offset  $\Delta\omega_0$  (difference between reference frequency and [scaled down] initial frequency of the VCO) is distinctly smaller than the pull-in range, typically less than 0.8 times the pull-in range. When  $\Delta\omega$  approaches the pull-in range  $\Delta\omega_P$ , the pull-in time  $T_P$  approaches infinity; thus, the formula cannot be used.

The analysis gives the results:

- For the passive lead-lag filter

$$T_P \approx \frac{\pi^2}{16} \frac{\Delta\omega_0^2}{\zeta\omega_n^3} \quad (3.72)$$

- For the active lead-lag filter

$$T_P \approx \frac{\pi^2}{16} \frac{\Delta\omega_0^2 K_a}{\zeta\omega_n^3} \quad (3.73)$$

■ For the active PI filter

$$T_P \approx \frac{\pi^2}{16} \frac{\Delta\omega_0^2}{\zeta\omega_n^3} \quad (3.74)$$

In these formulas  $\Delta\omega_0$  is the initial frequency offset  $\omega_1 - \omega_2'$  for  $t = 0$ . The quadratic and cubic terms in Eqs. (3.72) through (3.74) show that the pull-in process is highly nonlinear. The pull-in time  $T_P$  is normally much longer than the lock-in time  $T_L$ . This is demonstrated easily by a numerical example.

**Numerical Example** A second-order PLL having a passive lead-lag loop filter is assumed to operate at a center frequency  $f_0$  of 100 kHz. No down scaler is used, thus  $N = 1$ . Its natural frequency  $f_n = \omega_n/2\pi$  is 3 Hz, which is a very narrow-band system. The damping factor is chosen to be  $\zeta = 0.7$ . The loop gain  $K_0 K_d/N$  is assumed to be  $2\pi \cdot 1000 \text{ rad/s}^{-1}$ . We shall now calculate the lock-in time  $T_L$  and the pull-in time  $T_P$  for an initial frequency offset  $\Delta f_0$  of 30 Hz.

According to Eqs. (3.62) and (3.72), we get

$$T_L \approx 0.33 \text{ s}$$

$$T_P \approx 4.67 \text{ s}$$

$T_P$  is much larger than  $T_L$ .

**Phase detector type 2.** The pull-in range of a PLL using the EXOR phase detector can be calculated by performing a similar procedure as that used earlier with the multiplier phase detector. We assume the PLL is out of lock initially, that the VCO operates at its center frequency  $\omega_0$ , and that the initial offset  $\Delta\omega_0$  between reference frequency  $\omega_1$  and (down-scaled) VCO frequency  $\omega_0'$  is large. The signals  $u_1$  and  $u_2'$  can then be represented by

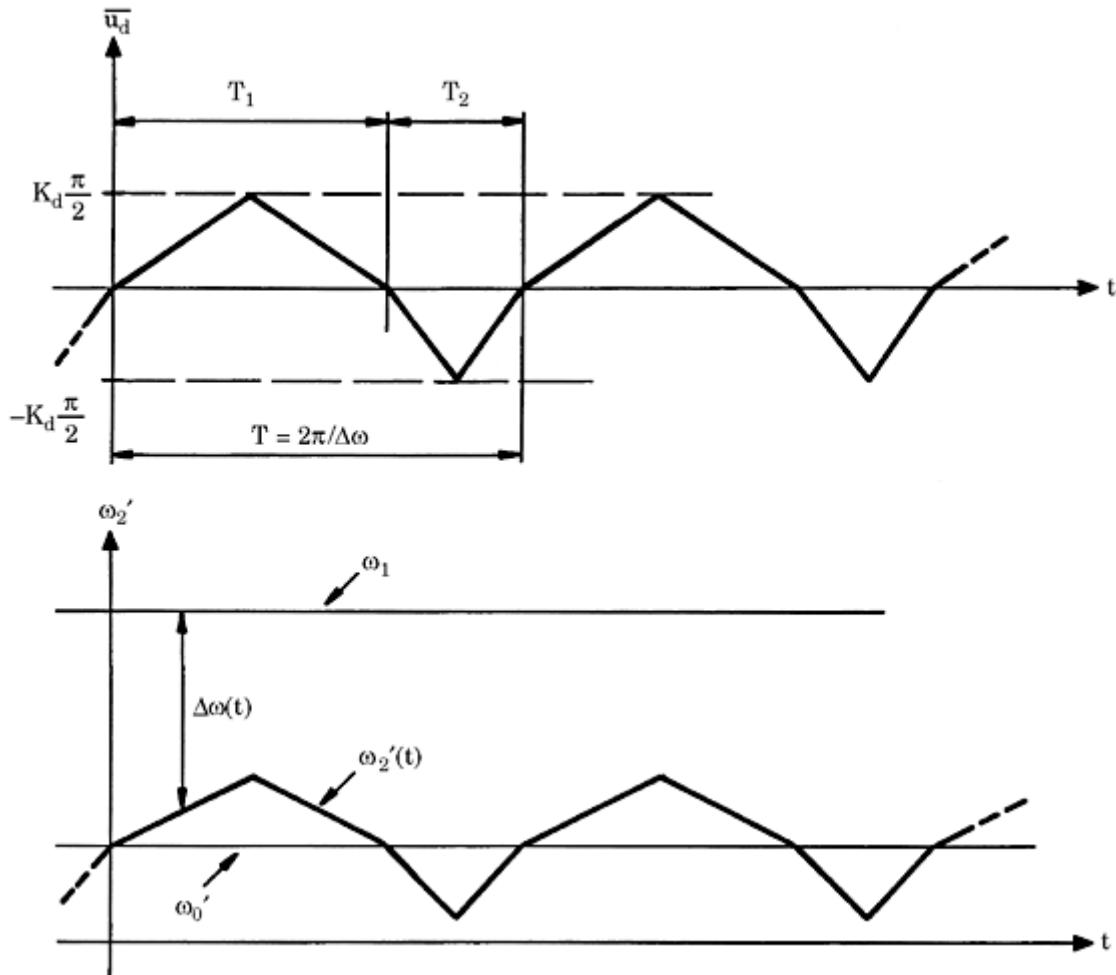
$$\begin{aligned} u_1(t) &= U_{10} \operatorname{rect}(\omega_0't + \Delta\omega_0 t) \\ u_2'(t) &= U_{20} \operatorname{rect}(\omega_0't) \end{aligned}$$

respectively, where  $U_{10}$  and  $U_{20}$  are the amplitudes of the square-wave signals. The phase error  $\theta_e$  is the difference of the phases of these two signals—that is

$$\theta_e(t) = \Delta\omega_0 \cdot t$$

which is a ramp function. The average output signal  $\overline{u_d}(t)$  is therefore a triangular signal, as shown in the upper trace of Fig. 3.16. (Let us discard for the moment the asymmetry of the waveform.) The output signal  $u_d(t)$  of the loop filter will be some fraction of the signal  $u_d(t)$  and will modulate the down-scaled instantaneous frequency  $\omega_2'(t)$  of the VCO, lower trace

in Fig. 3.16. If the triangular waveform was symmetrical (that is,  $T_1 = T_2$ ), the average frequency would remain constant and equal to  $\omega_0'$ . As Fig. 3.16 demonstrates, however, the frequency offset  $\Delta\omega(t)$  is not constant but is given by the difference between reference frequency  $\omega_1$  and the *instantaneous* (scaled-down) VCO frequency  $\omega_2'$ . Consequently,  $\Delta\omega(t)$  becomes smaller during the positive half-wave of the



**Figure 3.16** The figure explains the slow pull-in process of the PLL using the EXOR as phase detector. The upper trace shows the averaged output signal  $\bar{u}_d$  of the EXOR gate, while the lower trace displays the down-scaled instantaneous radian frequency  $\omega_2'(t)$ . The asymmetry of the waveforms is shown exaggerated. Because the duration of the positive half-wave of  $\bar{u}_d$  is larger than the negative, a DC offset is generated. If the loop gain of the PLL is sufficiently large, the loop is pulled in—in other words, the peak value of  $\omega_2'$  reaches  $\omega_1$  after some time.

$\bar{u}_d$  signal and larger during the negative half-wave. Therefore, the waveform of  $\Delta\omega(t)$  becomes asymmetrical, which is shown exaggerated in Fig. 3.16. When the  $\bar{u}_d$  waveform is asymmetrical, its mean value is no longer zero but becomes slightly positive. This causes the average frequency  $\omega_2$  of the VCO to be *pulled up*. Now different things can happen. If the loop gain (meaning, the product  $K_d K_0 F(0)/N$ ) is small, the VCO frequency is pulled up by a small amount only and stays stuck at some final value. If the loop gain is larger, however, the pull-in process becomes regenerative: the mean frequency is pulled up so much that the  $\bar{u}_d$  waveform becomes even more nonharmonic (that is, the ratio of  $T_1/T_2$  becomes significantly greater). This causes the mean  $\bar{\omega}_2$  frequency to increase even more, and so the scaled-down

VCO output frequency will be pulled up until it comes close to the reference frequency. Afterward, a locking process will take place. A pull-in process is initiated whenever the initial frequency offset  $\Delta\omega_0$  is smaller than the pull-in range  $\Delta\omega_P$ . The final results are given next (for details refer to [App. A](#)):

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

- Loop filter = passive lead-lag

$$\Delta\omega_P \approx \frac{\pi}{2} \sqrt{2\zeta\omega_n K_0 K_d / N - \omega_n^2} \quad \text{low-gain loops} \quad (3.75)$$

$$\Delta\omega_P \approx \frac{\pi}{\sqrt{2}} \sqrt{\zeta\omega_n K_0 K_d / N} \quad \text{high-gain loops}$$

- Loop filter = active lead-lag

$$\Delta\omega_P \approx \frac{\pi}{2} \sqrt{2\zeta\omega_n K_0 K_d / N - \omega_n^2 / K_a} \quad \text{low-gain loops} \quad (3.76)$$

$$\Delta\omega_P \approx \frac{\pi}{\sqrt{2}} \sqrt{\zeta\omega_n K_0 K_d / N} \quad \text{high-gain loops}$$

- Loop filter = active PI

$$\Delta\omega_P \rightarrow \infty \quad (3.77)$$

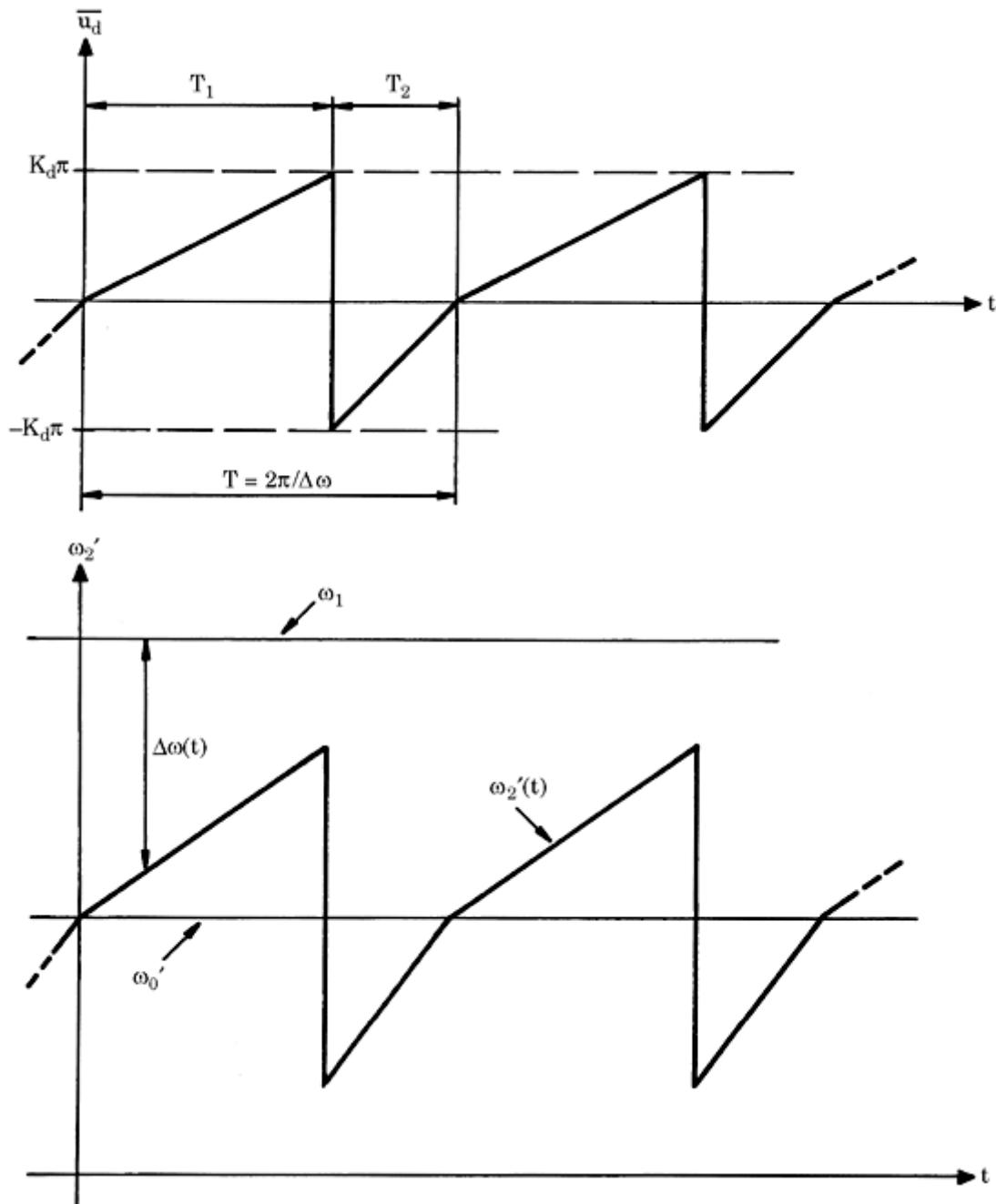
As demonstrated in [App. A](#), it is also possible to calculate an approximate value for the pull-in time  $T_P$ . The final result reads

$$\left. \begin{aligned} T_P &\approx \frac{4}{\pi^2} \frac{\Delta\omega_0^2}{\zeta\omega_n^3} && (\text{passive lead-lag filter}) \\ T_P &\approx \frac{4}{\pi^2} \frac{\Delta\omega_0^2 K_a}{\zeta\omega_n^3} && (\text{active lead-lag filter}) \\ T_P &\approx \frac{4}{\pi^2} \frac{\Delta\omega_0^2}{\zeta\omega_n^3} && (\text{active PI filter}) \end{aligned} \right\} \quad (3.78)$$

As we know, the pull-in time becomes infinite when the initial frequency offset equals the pull-in range. When the passive or active lead-lag filter is used, the approximation of [Eq. \(3.78\)](#) is valid only when  $\Delta\omega_0$  is markedly less than  $\Delta\omega_P$ . Computer simulations have shown that the approximation gives acceptable results when  $\Delta\omega_0$  is less than about 0.8  $\Delta\omega_P$ . (In practical terms, “acceptable” means the error of the predicted result is not larger than about 10 percent.)

**Phase detector type 3.** Now we analyze the pull-in process for the case where the JK-flipflop is used as a phase detector. Making the same assumptions as for the EXOR gate, the waveforms of the average  $\overline{u_d}(t)$  signal and the instantaneous (down-scaled) output frequency  $\omega_2'$  look like those drawn in [Fig. 3.17](#). Instead of triangular waves, we obtain sawtooth waves now. Performing an analogous computation like that done earlier, we get for the pull-in

range:



**Figure 3.17** The pull-in process of a PLL using the JK-flipflop as a phase detector. The upper trace shows the average phase detector output signal  $\bar{u}_d$ , while the lower trace shows the down-scaled frequency  $\omega_2'$  created by the VCO.

- For the passive lead-lag filter

$$\begin{aligned}\Delta\omega_P &\approx \pi \sqrt{2\zeta\omega_n K_0 K_d / N - \omega_n^2} && \text{low-gain loops} \\ \Delta\omega_P &\approx \pi \sqrt{2\zeta\omega_n K_0 K_d / N} && \text{high-gain loops}\end{aligned}\quad (3.79)$$

- For the active lead-lag filter

$$\begin{aligned}\Delta\omega_P &\approx \pi \sqrt{2\zeta\omega_n K_0 K_d / N - \omega_n^2 / K_a} && \text{low-gain loops} \\ \Delta\omega_P &\approx \pi \sqrt{2\zeta\omega_n K_0 K_d / N} && \text{high-gain loops}\end{aligned}\quad (3.80)$$

■ For the active PI filter

$$\Delta\omega_P \rightarrow \infty \quad (3.81)$$

The pull-in time becomes

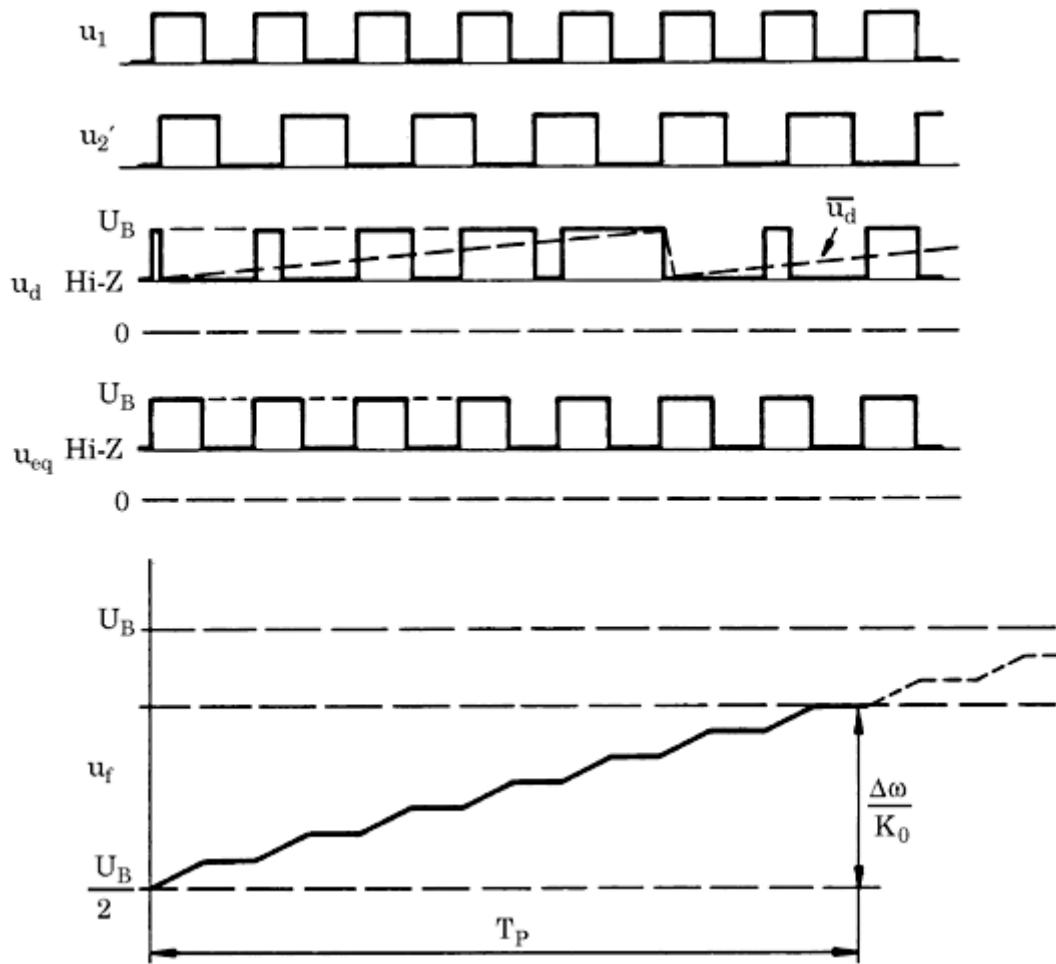
$$\left. \begin{aligned} T_P &\approx \frac{1}{\pi^2} \frac{\Delta\omega_0^2}{\zeta\omega_n^3} && (\text{passive lead-lag filter}) \\ T_P &\approx \frac{1}{\pi^2} \frac{\Delta\omega_0^2 K_a}{\zeta\omega_n^3} && (\text{active lead-lag filter}) \\ T_P &\approx \frac{1}{\pi^2} \frac{\Delta\omega_0^2}{\zeta\omega_n^3} && (\text{active PI filter}) \end{aligned} \right\} \quad (3.82)$$

When the passive or the active lead-filter is used, this formula gives acceptable results when  $\Delta\omega_0$  is less than about 0.8  $\Delta\omega_P$ .

**Phase detector type 4a.** When the PFD with voltage output is used as phase detector, we have to develop another model for the pull-in process because the dynamic behavior of the PFD differs greatly from that of all other phase detectors. As we have already seen, the output signal of the PFD depends on the phase error in the locked state of the PLL. When the PLL is out of lock, however, the output signal of the PFD depends on the frequency error (cf. Fig. 2.15). As explained in Sec. 2.4.4.1, the PFD has a tri-state output. We have seen in Sec. 3.9.1 that any loop filter cascaded with this type of PFD behaves as a real integrator (meaning it has a pole at  $s = 0$ ) when both flipflops of the PFD are in the 0 state, refer also to Eqs. (3.56) through (3.58). Hence, the DC gain of all loop filters is (at least theoretically) infinite, and the pull-in range  $\Delta\omega_P$  also becomes infinite. In practice, the pull-in range corresponds to the range of frequencies the VCO is able to generate.

The pull-in time remains finite, of course, so we need a suitable approximation for  $T_P$ . First, we will calculate the pull-in time  $T_P$  for the case where the passive lead-lag loop filter is used. We assume again that the PLL is initially out of lock and the VCO operates at its center frequency  $\omega_0$ . The input radian frequency  $\omega_1$  is assumed to be markedly higher than the down-scaled VCO output frequency  $\omega_0/N = \omega_0'$ . The initial radian frequency offset is denoted  $\Delta\omega_0$  with  $\Delta\omega_0 = \omega_1 - \omega_0'$ . This situation is sketched by the upper two waveforms in Fig. 3.18. As explained in Sec. 2.4.4.1 and Fig. 2.12, the output signal  $u_d$  of the PFD then toggles between the states 0 and 1. The third trace shows the waveform of  $u_d$ . It was assumed that the PFD is powered by a unipolar supply so the logical “high” level is  $U_B$ , while the “low” level is 0 V (ground). The centerline of the  $u_d$  signal represents the high-impedance state of the PFD (Hi-Z). The average  $u_d$  signal is drawn as a dashed line. It has the shape of a sawtooth signal. The average  $u_d$  signal is nothing else than the duty cycle of the PFD output. It

periodically ramps up from 0 to 1 and is a sawtooth function as well.



**Figure 3.18** The waveforms shown demonstrate the pull-in process of a PLL which uses the PFD. The frequency  $\omega_1$  of the input signal  $u_1$  is assumed to be higher than the (scaled-down) frequency  $\omega_2'$  of the VCO output signal. Consequently,  $u_d$  toggles between the states 0 and 1 (third trace). The average  $u_d$  signal has the same effect as an equivalent signal  $u_{eq}$  with a constant duty cycle of 50 percent, as explained in the text. This greatly facilitates the computation of the averaged loop filter output signal  $u_f$  (see the bottom trace).

Obviously, the average duty cycle of  $u_d$  is 50 percent. As Fig. 2.15 shows, the average duty cycle  $\delta$  varies very little with the ratio  $\omega_1/\omega_2'$  and can be considered constant for this analysis. Because the time constant  $\tau_1$  of the loop filter is much larger than the period of the  $u_1$  signal in Fig. 3.18, an equivalent signal  $u_{eq}$  (having a constant duty cycle of 50 percent) would have the same effect on the loop filter (this equivalent signal is also represented in Fig. 3.18). If the equivalent signal  $u_{eq}$  had a duty cycle of 100 percent, capacitor  $C_1$  of the loop filter would simply charge toward the supply voltage  $U_B$  with time constant  $\tau_1 + \tau_2 = (R_1 + R_2) C_1$ , because  $C_1$  is charged through the series connection of resistors  $R_1$  and  $R_2$  (compare Fig. 2.17a for symbol definitions). Because the duty cycle is only 50 percent, however, the

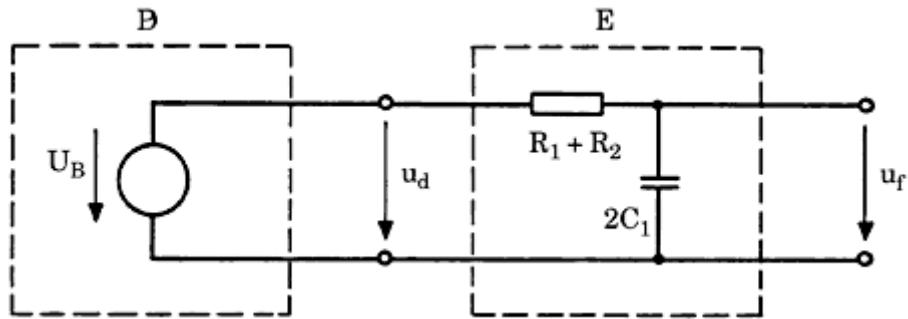
capacitor needs twice as much time to charge. Therefore, the loop filter acts like a simple  $RC$  filter whose time constant is not  $\tau_1 + \tau_2$  but  $2(\tau_1 + \tau_2)$ . The equivalent model of Fig. 3.19 can now be used to compute the signal  $u_f$  across capacitor  $C_1$ . Since the VCO of the PLL was assumed to operate at its center frequency  $\omega_0$  at the start of the pull-in process, the initial

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).

Copyright ©2004 The McGraw-Hill Companies. All rights reserved.

Any use is subject to the Terms of Use as given at the website.



**Figure 3.19** Charging of capacitor  $C_1$  in the loop filter is calculated by this equivalent model.  
Detailed explanations are in the text.

value of  $u_f$  is  $U_B/2$ . Consequently, the capacitor will try to charge from  $U_B/2$  to  $U_B$  during the pull-in process. The actual source voltage for the  $RC$  filter in Fig. 3.19 is therefore  $U_B$ .

To get locked, the VCO must create an output frequency that is offset from the center frequency  $\omega_0$  by the amount  $N \Delta\omega_0$ . To obtain that frequency the voltage on capacitor  $C_1$  must be increased by  $N \cdot \Delta\omega_0/K_0$  [compare with Eq. (3.4)]. The pull-in time  $T_P$  now is simply the time after which the voltage on capacitor  $C_1$  has reached that level. For the passive lead-lag filter, the calculation yields

■ loop filter = passive lead-lag

$$T_P \approx 2(\tau_1 + \tau_2) \ln \frac{1}{1 - \frac{2N\Delta\omega_0}{U_B K_0}} \quad (3.83)$$

where  $\Delta\omega_0$  is the initial frequency offset,  $\Delta\omega_0 = \omega_1 - \omega_0'$ . This formula has been derived under the premise that the PFD is driven by a unipolar power supply. In the most general case, the PFD could be driven from a bipolar supply, where the positive and negative supply voltages are  $U_{B+}$  and  $U_{B-}$ , respectively. Furthermore, the output signal of the PFD output signal could be clipped at the saturation levels  $U_{\text{sat}+}$  (positive) and  $U_{\text{sat}-}$  (negative), respectively. To account for clipping, we simply have to replace  $U_B/2$  in Eq. (3.83) by  $(U_{\text{sat}+} - U_{\text{sat}-})/2$ .

For the active lead-lag filter, a similar analysis yields the result

■ loop filter = active lead-lag

$$T_P \approx 2\tau_1 \ln \frac{1}{1 - \frac{2N\Delta\omega_0}{U_B K_0 K_a}} \quad (3.84)$$

An analogous computation can be performed for the case where the active PI is used:

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

■ loop filter = active PI

$$T_P \approx \frac{4\tau_1 \Delta\omega_0 N}{K_0 U_B} \quad (3.85)$$

It is worth considering a major difference of the pull-in processes for different types of phase detectors. If the phase detector is an EXOR gate, the instantaneous frequency of the VCO is modulated in both directions around its average value, as seen in the lower trace of Fig. 3.16. This is to be expected, for the  $u_d$  signal is an AC signal. Provided a pull-in process starts, the frequency of the VCO is slowly “pumped up,” as has been shown in Fig. 3.15 for the multiplier phase detector. A similar “pumping” is observed when the phase detector is a JK-flipflop (cf. Fig. 3.17). No “pumping” occurs, however, when the PFD is used. Since the driving signal for the loop filter is unipolar here (cf. Fig. 3.19), charge is “pumped” into the filter capacitor *in one direction* only, so that the frequency of the VCO is moved in the “right” way at any time. The instantaneous frequency of the VCO approaches the final value from one side only. When the pull-in process is completed, a lock-in process follows. Only then does the output frequency perform a damped oscillation; it slightly overshoots the final value and settles after the transient has died out. We will have a closer look at these phenomena when we perform computer simulations in Chap. 10.

**Phase detector type 4b.** The current output PFD is almost always combined with a passive lead-lag loop filter (cf. Fig. 2.17b), hence we can restrict discussion on this case. Because a current-driven lead-lag filter has a pole at  $s = 0$  and consequently behaves like a real integrator, the pull-in range becomes infinite, like in the case of the voltage-driven PFD. To compute the pull-in time  $T_P$ , we will use a model similar to that used for PFD type 4a (refer to Fig. 3.19). For the voltage output PFD, the PFD output signal  $u_d$  is ramped up in a sawtooth-like manner, as illustrated by the third trace in Fig. 3.18. For the current output PFD, this signal must be replaced by a current output  $i_d$ . The average value of  $i_d$  also shows up as a sawtooth waveform varying from 0 to  $I_P/2$ , where  $I_P$  is the amplitude of the current source in Fig. 2.16, which was defined in Eq. (2.26). Because  $I_P = K_P 2\pi$  [Eq. (2.27)] the time varying current source can be replaced by an equivalent current source  $I_{eq} = K_P \pi$ . This current source now charges capacitor  $C_1$  of the loop filter. The pull-in time  $T_P$  is the time required to change the voltage on capacitor  $C_1$  such that the (scaled-down) frequency of the VCO  $\omega_2'$  becomes equal to the frequency  $\omega_1$  of the input signal. An analogous analysis of the pull-in process yields the pull-in time

$$T_P = \Delta\omega_0 \cdot \frac{NC_1}{K_P K_0 \pi} \quad (3.86)$$

**The truth about “infinite pull-in range”.** Theory suggests that the pull-in range becomes infinite whenever the loop filter is an active PI filter—that is, a filter having a pole at  $s = 0$  (or in other words, a filter having “infinite gain” at DC). As we already

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

have seen, the pull-in range cannot be larger than the range of frequencies the VCO is able to create. However, there is another, even more stringent, restriction. When phase detector type 1 or 2 is used, the PLL can false-lock onto a harmonic of the reference frequency. To explain that phenomenon, we'll use two examples.

**Example 1** Consider a PLL with type 1 phase detector (multiplier) and no down scaler. Let the center frequency be  $f_0 = 100$  kHz. Assume that the reference frequency is  $f_1 = 100$  kHz initially. Now the reference frequency suddenly jumps to 50 kHz. We would expect the VCO to pull down its frequency to 50 kHz, but it remains locked at 100 kHz. What happened? With  $f_1 = 100$  kHz and  $f_2 = 50$  kHz the multiplier generates the sum and difference of these two frequencies—for instance, 150 kHz and 50 kHz. The frequency of the VCO is thus modulated by a 50 kHz and a 150 kHz component. The frequency modulation generated by the 50 kHz component creates sidebands at 50 kHz and 150 kHz. The lower sideband (50 kHz) has exactly the reference frequency, which causes the phase detector to output a DC signal, which depends on the phase shift between these two 50 kHz signals. The phase detector therefore “believes” that the PLL is perfectly locked, but in effect it is locked to twice the reference frequency.

**Example 2** Consider a PLL with type 2 phase detector (EXOR) and no down scaler. Again, the center frequency is  $f_0 = 100$  kHz. The reference frequency  $f_1$  is assumed to be 100 kHz initially. Now  $f_1$  suddenly jumps down to 40 kHz. We would expect the VCO to pull down its frequency to 40 kHz as well, but it ramps up to 120 kHz and stays locked there! What happened? The PLL locked onto the third harmonic of the reference frequency. In effect, the reference signal, which is assumed to be a symmetrical square wave has odd harmonics—in other words, third, fifth, and so on. The third harmonic is 120 kHz. When the VCO oscillates at 120 kHz, the EXOR phase detector thinks the PLL is perfectly locked. Indeed, it is locked, but on the *wrong* frequency.

It shows up that the full pull-in range can only be realized when either type 3 or type 4 phase detectors are chosen. Because phase detector type 4 is not only phase- but also frequency-sensitive, the pull-in process is much faster if type 4 is selected.

### Pull-out range $\Delta\omega_{PO}$

**Phase detector type 1.** The pull-out range is, by definition, the frequency step that causes a lock-out if applied to the reference input of the PLL. In the mechanical analogy of Fig. 3.7, the pull-out frequency corresponds to the weight that causes the pendulum to tip over if the weight is suddenly dropped onto the platform.

An exact calculation of the pull-out range is not possible for the linear PLL. However, simulations on an analog computer<sup>6</sup> have led to an approximation:

$$\Delta\omega_{PO} \approx 1.8\omega_n(\zeta + 1) \quad (3.87)$$

In most practical cases, the pull-out range is between the lock range and the pull-in range

$$\Delta\omega_L < \Delta\omega_{PO} < \Delta\omega_P \quad (3.88)$$

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

If, in an FM system, the PLL is pulled out by too large a frequency step, we can expect that PLL to return to stable operation—by means of a relatively slow pull-in process—provided the frequency offset  $\Delta\omega = \omega_1 - \omega_0'$  is smaller than  $\Delta\omega_P$ . If the corresponding pull-in time is considered to be too long, the peak frequency deviation  $\Delta\omega$  must be confined to the lock range  $\Delta\omega_L$ .

**Phase detector type 2.** When the EXOR phase detector is chosen, the pull-out range also is the frequency step that causes the pendulum in the model of Fig. 3.7 to tip over. With the EXOR phase detector, the average output signal  $\bar{u}_d$  versus phase error is a triangular function, as shown in Fig. 2.7. Because this is a nonlinear function, it is not possible to calculate explicitly the pull-out frequency. Using the PLL design program distributed with this book, the pull-out range was determined by simulation, using damping factors in the range  $0.1 < \zeta < 3$ . Then, a least-squares fit gave the approximation

$$\Delta\omega_{PO} \approx 2.46\omega_n(\zeta + 0.65) \quad (3.89)$$

As could be expected, this result is not far from that for phase detector type 1.

**Phase detector type 3.** A different procedure is used to compute the pull-out range of the PLLs using a JK-flipflop or a PFD as phase detector. In the case of the JK-flipflop, the pull-out range is the frequency step causing the peak phase error to exceed  $\pi$ . Because the average output signal  $\bar{u}_d$  of the JK-flipflop actually is *linear* in the range  $-\pi < \theta_e < \pi$ , the pull-out range can be computed explicitly. Using the linear model of the PLL (Fig. 3.1), phase error  $\theta_e$  is calculated for a frequency step  $\Delta\omega$  applied to the reference input. The result is a damped oscillation.<sup>1</sup> Using the rules of differential calculus, it is straightforward to calculate the maximum of the phase error. From there, it is quite easy to calculate the size of the frequency step that leads to a peak phase error of  $\pi$ . Assuming the PLL is a high-gain loop, we get

$$\Delta\omega_{PO} = \begin{cases} \pi\omega_n \exp\left(\frac{\zeta}{\sqrt{1-\zeta^2}} \tan^{-1} \frac{1-\zeta^2}{\zeta}\right), & \zeta < 1 \\ \pi\omega_n e, & \zeta = 1 \\ \pi\omega_n \exp\left(\frac{\zeta}{\sqrt{\zeta^2-1}} \tanh^{-1} \frac{\zeta^2-1}{\zeta}\right), & \zeta > 1 \end{cases} \quad (3.90)$$

If  $\Delta\omega_{PO}$  is plotted against  $\zeta$ , we notice that the curve becomes rather flat and could easily be replaced by a linear function. This would ease the computation of the pull-out range considerably, because tables for inverse hyperbolic tangent, for example, are not always at hand. A least-squares fit performed with Eq. (3.90) gave the approximation

$$\Delta\omega_{PO} \approx 5.78\omega_n(\zeta + 0.5) \quad (3.91)$$

**Phase detector type 4.** The following analysis is valid for both voltage- and current-output PFDs, because the normalized phase-transfer function  $H(s)$  is the same

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

for both variants. In the case of the PFD, the pull-out range is the frequency step causing the peak phase error to exceed  $2\pi$ . Because the average output signal  $\bar{u}_d$  of the JK-flipflop actually is *linear* in the range  $-2\pi < \theta_e < 2\pi$ , the pull-out range can be computed explicitly, as done for phase detector type 3. An analogous computation yields

$$\Delta\omega_{PO} = \begin{cases} 2\pi\omega_n \exp\left(\frac{\zeta}{\sqrt{1-\zeta^2}} \tan^{-1} \frac{1-\zeta^2}{\zeta}\right), & \zeta < 1 \\ 2\pi\omega_n e, & \zeta = 1 \\ 2\pi\omega_n \exp\left(\frac{\zeta}{\sqrt{\zeta^2-1}} \tanh^{-1} \frac{\zeta^2-1}{\zeta}\right), & \zeta > 1 \end{cases} \quad (3.92)$$

Here, the least-squares fit gives the linear approximation

$$\Delta\omega_{PO} \approx 11.55\omega_n(\zeta + 0.5) \quad (3.93)$$

To ease the design of mixed-signal PLLs, the equations derived in Sec. 3.9 are summarized in five tables, one table for each type of phase detector. Table 3.1 lists the formulas for the most important key parameters for PLLs using phase detector type 1, Table 3.2 contains the corresponding equations for the EXOR

**TABLE 3.1 Summary of Parameters of the Mixed-Signal PLL, Phase Detector Type 1 (Multiplier)**

Key parameter	Loop filter	
	Passive lead-lag	Active lead-lag
Natural frequency $\omega_n$	$\sqrt{\frac{K_0 K_d}{N(\tau_1 + \tau_2)}}$	$\sqrt{\frac{K_0 K_d K_a}{N \tau_1}}$
Damping factor $\zeta$	$\frac{\omega_n}{2} \left( \tau_2 + \frac{N}{K_0 K_d} \right)$	$\frac{\omega_n}{2} \left( \tau_2 + \frac{N}{K_0 K_d K_a} \right)$
Hold range $\Delta\omega_H$	$K_0 K_d / N$	$K_0 K_d K_a / N$
Lock range $\Delta\omega_L$		$\approx 2\zeta\omega_n$
Lock time $T_L$		$\approx \frac{2\pi}{\omega_n}$
Pull-in range $\Delta\omega_P$	Low-gain loops $\frac{4}{\pi} \sqrt{2\zeta\omega_n K_0 K_d / N - \omega_n^2}$	Low-gain loops $\frac{4}{\pi} \sqrt{2\zeta\omega_n K_0 K_d K_a / N - \frac{\omega_n^2}{K_a}}$
	High-gain loops $\frac{4\sqrt{2}}{\pi} \sqrt{\zeta\omega_n K_0 K_d / N}$	High-gain loops $\frac{4\sqrt{2}}{\pi} \sqrt{\zeta\omega_n K_0 K_d / N}$
Pull-in time $T_P$	$\approx \frac{\pi^2}{16} \frac{\Delta\omega_0^2}{\zeta\omega_n^3}$	$\approx \frac{\pi^2}{16} \frac{\Delta\omega_0^2 K_a}{\zeta\omega_n^3}$
Pull-out range $\Delta\omega_{PO}$		$\approx 1.8\omega_n(\zeta + 1)$

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
 Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
 Any use is subject to the Terms of Use as given at the website.

**TABLE 3.2 Summary of Parameters of the Mixed-Signal PLL, Phase Detector Type 2 (EXOR)**

Key parameter	Loop filter	
	Passive lead-lag	Active lead-lag
Natural frequency $\omega_n$	$\sqrt{\frac{K_0 K_d}{N(\tau_1 + \tau_2)}}$	$\sqrt{\frac{K_0 K_d K_a}{N \tau_1}}$
Damping factor $\zeta$	$\frac{\omega_n}{2} \left( \tau_2 + \frac{N}{K_0 K_d} \right)$	$\frac{\omega_n}{2} \left( \tau_2 + \frac{N}{K_0 K_d K_a} \right)$
Hold range $\Delta\omega_H$	$\frac{K_0 K_d \pi/2}{N}$	$\frac{K_0 K_d K_a \pi/2}{N}$
Lock range $\Delta\omega_L$		$\approx \pi \zeta \omega_n$
Lock time $T_L$		$\approx \frac{2\pi}{\omega_n}$
Pull-in range $\Delta\omega_P$	Low-gain loops $\frac{\pi}{2} \sqrt{2\zeta\omega_n K_0 K_d / N - \omega_n^2}$ High-gain loops $\frac{\pi}{\sqrt{2}} \sqrt{\zeta\omega_n K_0 K_d / N}$	Low-gain loops $\frac{\pi}{2} \sqrt{2\zeta\omega_n K_0 K_d K_a / N - \frac{\omega_n^2}{K_a}}$ High-gain loops $\frac{\pi}{\sqrt{2}} \sqrt{\zeta\omega_n K_0 K_d / N}$
Pull-in time $T_P$	$\approx \frac{4}{\pi^2} \frac{\Delta\omega_0^2}{\zeta\omega_n^3}$	$\approx \frac{4}{\pi^2} \frac{\Delta\omega_0^2 K_a}{\zeta\omega_n^3}$
Pull-out range $\Delta\omega_{PO}$		$\approx 2.46\omega_n(\zeta + 0.65)$

phase detector, [Table 3.3](#) summarizes the parameters of the Mixed-Signal PLL using phase detector type 3, [Table 3.4](#) summarizes the parameters of the mixed-signal PLL for phase detector type 4a (PFD with voltage output). [Table 3.5](#) summarizes the parameters of the mixed-signal PLL using phase detector type 4b.

## Optimizing the Lock Process

The lock time  $T_L$  is one of the most important parameters of a PLL. In most PLL applications, we want the PLL to acquire lock fast—for example, within 20  $\mu\text{s}$ . As shown in [Eq. \(3.62\)](#), the lock time is inversely proportional to the natural frequency  $\omega_n$ , and because the bandwidth of the PLL [cf. the formula for  $\omega_{3db}$  in [Eq. \(3.21\)](#)] is proportional to natural frequency, fast lock

automatically means large PLL bandwidth. As we will see in [Chap. 4](#) when discussing noise properties of PLLs, the susceptibility to noise superimposed to the input signal also increases with PLL bandwidth. Thus, to get low noise at the PLL output, we would prefer lower bandwidth.

**TABLE 3.3 Summary of Parameters of the Mixed-Signal PLL, Phase Detector Type 3 (JK-Flipflop)**

Key parameter	Loop filter	
	Passive lead-lag	Active lead-lag
Natural frequency $\omega_n$	$\sqrt{\frac{K_0 K_d}{N(\tau_1 + \tau_2)}}$	$\sqrt{\frac{K_0 K_d K_a}{N \tau_1}}$
Damping factor $\zeta$	$\frac{\omega_n}{2} \left( \tau_2 + \frac{N}{K_0 K_d} \right)$	$\frac{\omega_n}{2} \left( \tau_2 + \frac{N}{K_0 K_d K_a} \right)$
Hold range $\Delta\omega_H$	$\frac{K_0 K_d \pi}{N}$	$\frac{K_0 K_d K_a \pi}{N}$
Lock range $\Delta\omega_L$		$\approx 2\pi\zeta\omega_n$
Lock time $T_L$		$\approx \frac{2\pi}{\omega_n}$
Pull-in range $\Delta\omega_P$	Low-gain loops $\pi \sqrt{2\zeta\omega_n K_0 K_d / N - \omega_n^2}$	Low-gain loops $\pi \sqrt{2\zeta\omega_n K_0 K_d K_a / N - \frac{\omega_n^2}{K_a}}$
	High-gain loops $\pi \sqrt{2\zeta\omega_n K_0 K_d / N}$	High-gain loops $\pi \sqrt{2\zeta\omega_n K_0 K_d / N}$
Pull-in time $T_P$	$\approx \frac{1}{\pi^2} \frac{\Delta\omega_0^2}{\zeta\omega_n^3}$	$\approx \frac{1}{\pi^2} \frac{\Delta\omega_0^2 K_a}{\zeta\omega_n^3}$
Pull-out range $\Delta\omega_{PO}$		$\approx 5.78\omega_n(\zeta + 0.5)$

This dilemma can be fixed by using the so-called fastlock technique, which will be discussed in [Sec. 3.10.1](#).

## Fastlock techniques

To achieve fastlock, the bandwidth of the PLL is made large when it has not yet locked. As soon as lock is detected, the bandwidth is reduced to a smaller value. The fastlock technique will be explained by the example of a PLL that uses a current output PFD. (Such an application is found, for example, in the PLL frequency synthesizer chip type LMX 2470 manufactured by National Semiconductors.<sup>54</sup>) According to [Eq. \(3.23\)](#) for this type of PLL, the natural frequency  $\omega_n$  and damping factor  $\zeta$  are given by

$$\omega_n^2 = \frac{K_p K_0}{N C_1}, \quad \zeta = \frac{\omega_n \tau_2}{2}$$

During fastlock in many PLL frequency synthesizer ICs, the detector gain  $K_p$  is switched to a higher value—for instance,  $K_{PFL} = K_{FL} \cdot K_p$ , where  $K_{PFL}$  is the

**TABLE 3.4 Summary of Parameters of the Mixed-Signal PLL, Phase Detector Type 4a  
(PFD with Voltage Output)**

Key parameter	Loop filter	
	Passive lead-lag	Active lead-lag
Natural frequency $\omega_n$	$\sqrt{\frac{K_0 K_d}{N(\tau_1 + \tau_2)}}$	$\sqrt{\frac{K_0 K_d K_a}{N\tau_1}}$
Damping factor $\zeta$		$\frac{\omega_n \tau_2}{2}$
Hold range $\Delta\omega_H$		$\infty$
Lock range $\Delta\omega_L$		$\approx 4\pi\zeta\omega_n$
Lock time $T_L$		$\approx \frac{2\pi}{\omega_n}$
Pull-in range $\Delta\omega_P$		$\infty$
Pull-in time $T_P$	$2(\tau_1 + \tau_2) \ln \frac{1}{1 - \frac{2N\Delta\omega_0}{U_B K_0}}$	$2\tau_1 \ln \frac{1}{1 - \frac{2N\Delta\omega_0}{U_B K_0 K_a}}$
Pull-out range $\Delta\omega_{PO}$		$\approx 11.55\omega_n(\zeta + 0.5)$

increased detector gain during fastlock and  $K_{FL}$  is a multiplying factor. Detector gain is increased simply by switching the current sources within the phase detector (cf. Fig. 2.16) to a higher value.  $\omega_n$  varies with the square root of  $K_P$ ; hence, to double PLL bandwidth we would have to increase  $K_P$  by a factor of 4. Because the damping factor  $\zeta$  increases proportional to natural frequency  $\omega_n$ ,  $\zeta$  would be doubled during fastlock in the last example. This is not desirable,

**TABLE 3.5 Summary of Parameters of the Mixed-Signal PLL, Phase Detector Type 4b  
(PFD with Current Output)**

Key parameter	Loop filter	
	Passive lead-lag	
Natural frequency $\omega_n$	$\omega_n^2 = \frac{K_p K_0}{N C_1}$	
Damping factor $\zeta$	$\zeta = \frac{\omega_n \tau_2}{2}$	
Hold range $\Delta\omega_H$		$\infty$

Lock range  $\Delta\omega_L$

$$\Delta\omega_L \approx 4\pi\zeta\omega_n$$

Lock time  $T_L$

$$\approx \frac{2\pi}{\omega_n}$$

Pull-in range  $\Delta\omega_P$

$\infty$

Pull-in time  $T_P$

$$T_P = \Delta\omega_0 \cdot \frac{NC_1}{K_p K_0 \pi}$$

Pull-out range  $\Delta\omega_{PO}$

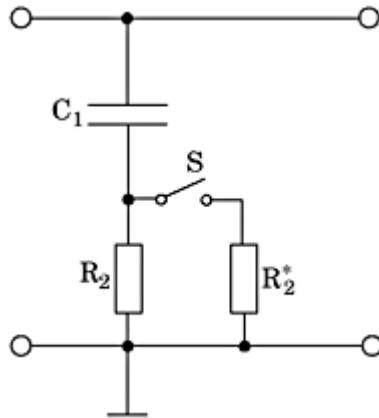
$$\approx 11.55\omega_n(\zeta + 0.5)$$

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).

Copyright ©2004 The McGraw-Hill Companies. All rights reserved.

Any use is subject to the Terms of Use as given at the website.



**Figure 3.20** A passive lead-lag loop filter with fastlock capability.

because for large damping factors the transient response of the PLL becomes sluggish. In order to keep  $\zeta$  unchanged during fastlock, we will have to switch an additional resistor  $R_2^*$  into the loop filter, as shown in Fig. 3.20.

To get the same damping factor  $\zeta$  during fastlock, resistor  $R_2^*$  is switched parallel to resistor  $R_2$  by analog switch S. If the detector gain is increased by a factor  $K_{FL}$  during fastlock, the parallel connection of  $R_2$  and  $R_2^*$  must now result in a resistor

$$R_2 \parallel R_2^* = R_2 / \sqrt{K_{FL}}$$

For the preceding example ( $K_{FL} = 4$ ),  $R_2^*$  would have to be chosen the same as  $R_2$ . To switch the current sources in the PFD and the parallel resistor in the loop filter, we need a signal telling us whether or not the PLL has acquired lock. Such a device is called an in-lock detector and will be discussed in Sec. 3.11.

## Cycle slip reduction (CSR)

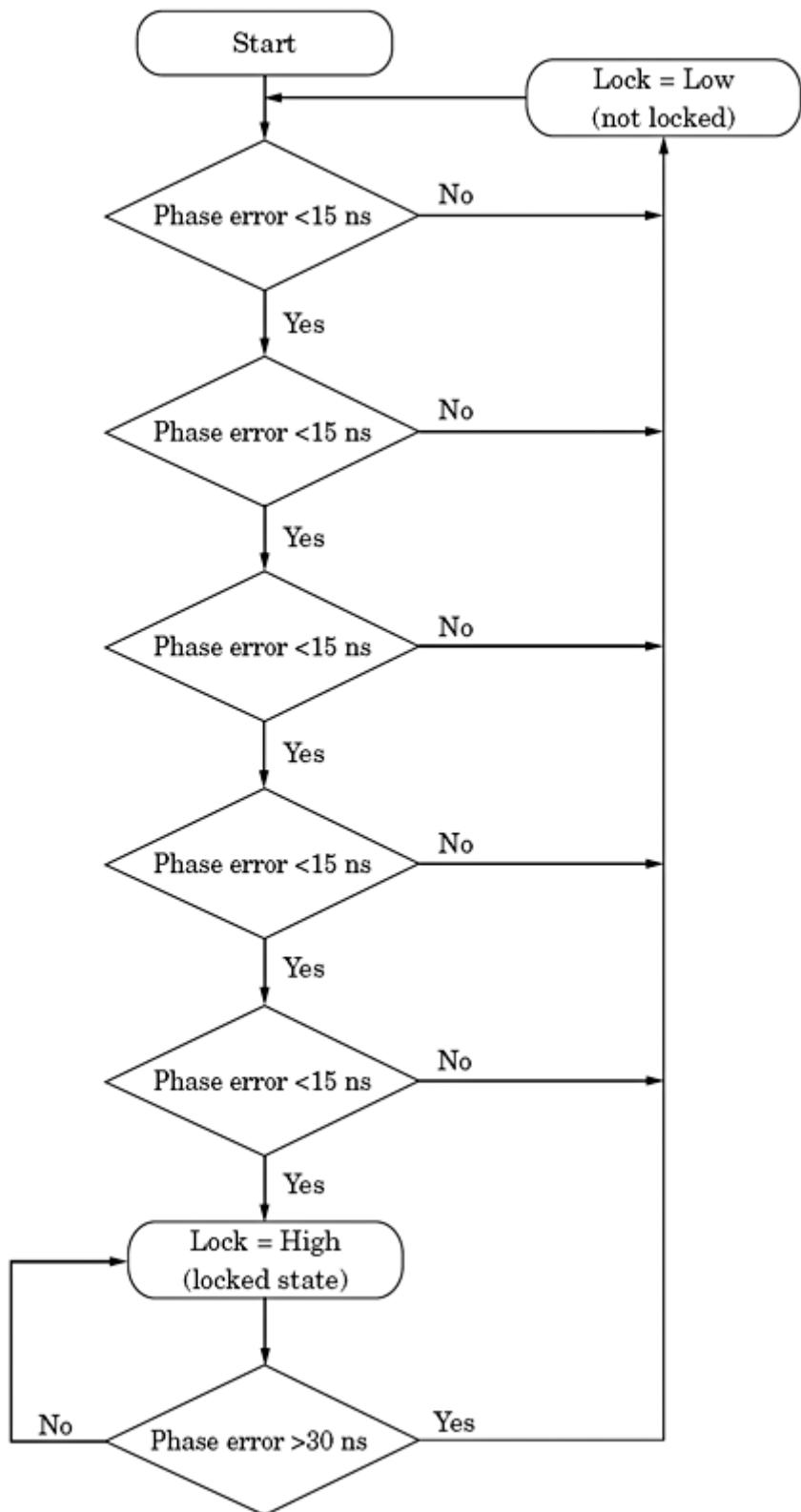
As will be shown in Chaps. 6 and 7, frequency synthesis is one of the major applications of PLLs. A frequency synthesizer must be able to switch from one output frequency to another in a relatively short time. When it does, it often happens that the PLL momentarily loses lock before settling at a new output frequency. When the frequency of the  $u_1$  signal is higher than that of the  $u_2'$  signal, the latter may “miss” a number of cycles when pulling to the new frequency; we say that some cycles are “slipped.” When the PLL is required to switch to a lower output frequency, the reverse is true: now the  $u_2'$  signal shows up with some

“extra” cycles. Cycle slipping can be reduced by a simple technique called CSR (cycle slip reduction). We will explain CSR using a simple example. Assume that the PLL is initially locked and is then requested to increase its output frequency by more than the lock range, so the loop will temporarily lose its lock. The phase error will therefore increase from zero in a positive direction. When the phase error exceeds the value  $2\pi$  after say four reference cycles (cycles of the input signal  $u_1$ ), the PLL will unlock after four cycles. When using CSR, we

reduce the so-called comparison frequency  $f_{\text{comp}}$  to a fraction of the reference frequency  $f_{\text{ref}}$ :

$$f_{\text{comp}} = K_{\text{CSR}} f_{\text{ref}}$$

where  $K_{CSR}$  can take the values 1, 1/2, 1/3, and so on. Actually, we do not evaluate the phase error in every reference cycle, only at a lower frequency  $f_{comp}$ . When we set  $K_{CSR} = 1/4$ , for example, and the phase error attains the value  $2\pi$  in the fourth reference cycle, the phase error referred to the reduced comparison frequency becomes four times less—in other words, only  $\pi/2$ . Under these conditions,



**Figure 3.21** Operating principle of an in-lock detector.

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

the loop will not lock out in the fifth cycle. Because the actual phase error is only a fraction  $K_{CSR}$  of the phase error that would be seen without CSR, this reduces the detector gain to

$$K_{P,CSR} = K_P K_{CSR}$$

This implies that the loop gain is decreased, and the natural frequency  $\omega_n$  is reduced to  $\omega_n \sqrt{K_{CSR}}$ . Consequently, the loop will lock slower by a factor  $1/\sqrt{K_{CSR}}$ .

To avoid this drawback, we can combine CSR with fastlock. When doing so, the resulting detector gain becomes

$$K_{P,comb} = K_{FL} \cdot K_{CSR} \cdot K_P$$

This equation tells us that the PLL locks faster by a factor of  $\sqrt{K_{FL} \cdot K_{CSR}}$ . This implies that the product  $K_{FL} \cdot K_{CSR}$  must be made larger than 1 in order to get faster acquisition. We could, for instance, choose  $K_{FL} = 16$  and  $K_{CSR} = 1/4$  to have  $K_{P,comb}$  four times larger than the initial  $K_P$ . When using fastlock in combination with CSR, we would have to select resistor  $R_2^*$  in Fig. 3.20 such that the parallel connection of  $R_2$  and  $R_2^*$  becomes  $R_2/\sqrt{K_{P,comb}} = R_2/\sqrt{K_{FL} \cdot K_{CSR}}$ .

## In-Lock detectors

We saw in Sec. 3.10 that switching the filter resistor  $R_2^*$  and the comparison frequency in frequency synthesizers requires an *in-lock detector*. An in-lock detector is a logical circuit having a binary output that tells us whether or not the PLL has acquired lock. Many different schemes are used to implement in-lock detectors. One simple method is depicted in Fig. 3.21.

This principle is utilized in National's frequency synthesizer IC type LMX2470.<sup>54</sup> To check if the PLL is locked, the in-lock detector measures the time difference between two consecutive positive transitions of the  $u_1$  and  $u_2'$  signals (input signals to the phase detector). The loop is decided to be in lock when this time difference is less than 15 ns in five consecutive reference cycles (cycles of the  $u_1$  signal). Whenever a time difference larger than 15 ns is found, the logical output of the in-lock detector is set low.

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

# PLL Performance in the Presence of Noise

Noise is an extremely important issue in the wide field of PLL applications. Noise has two entirely different aspects, however. In a first category of applications, the PLL is used to extract weak signals from an extremely noisy environment. The PLL has been very effective in space communications where it is used to detect signals buried in noise. In this situation, the input signal of the PLL is a low bandwidth signal with large superimposed broadband noise. In a quite different category of applications—for instance, frequency synthesizers—the PLL is required to generate an output signal with high spectral purity. Here, the PLL does not “receive” a noisy input signal but generates the noise itself. As we will see in Chaps. 6 and 7, the output signal of such PLLs contains a broadband phase noise spectrum. Outside of that, it may show spurious frequencies (unwanted spectral lines), something also called “spurs” or “tones,” as mentioned earlier in the book. In integer- $N$  frequency synthesizers, these spurs are generated by the phase detector (more about this in [Chap. 6](#)). In fractional-N frequency synthesizers, additional spurs are created by the sigma-delta modulators that are used to switch the divider ratio of the down scaler from one value to another. This will be discussed in greater detail in [Chap. 7](#).

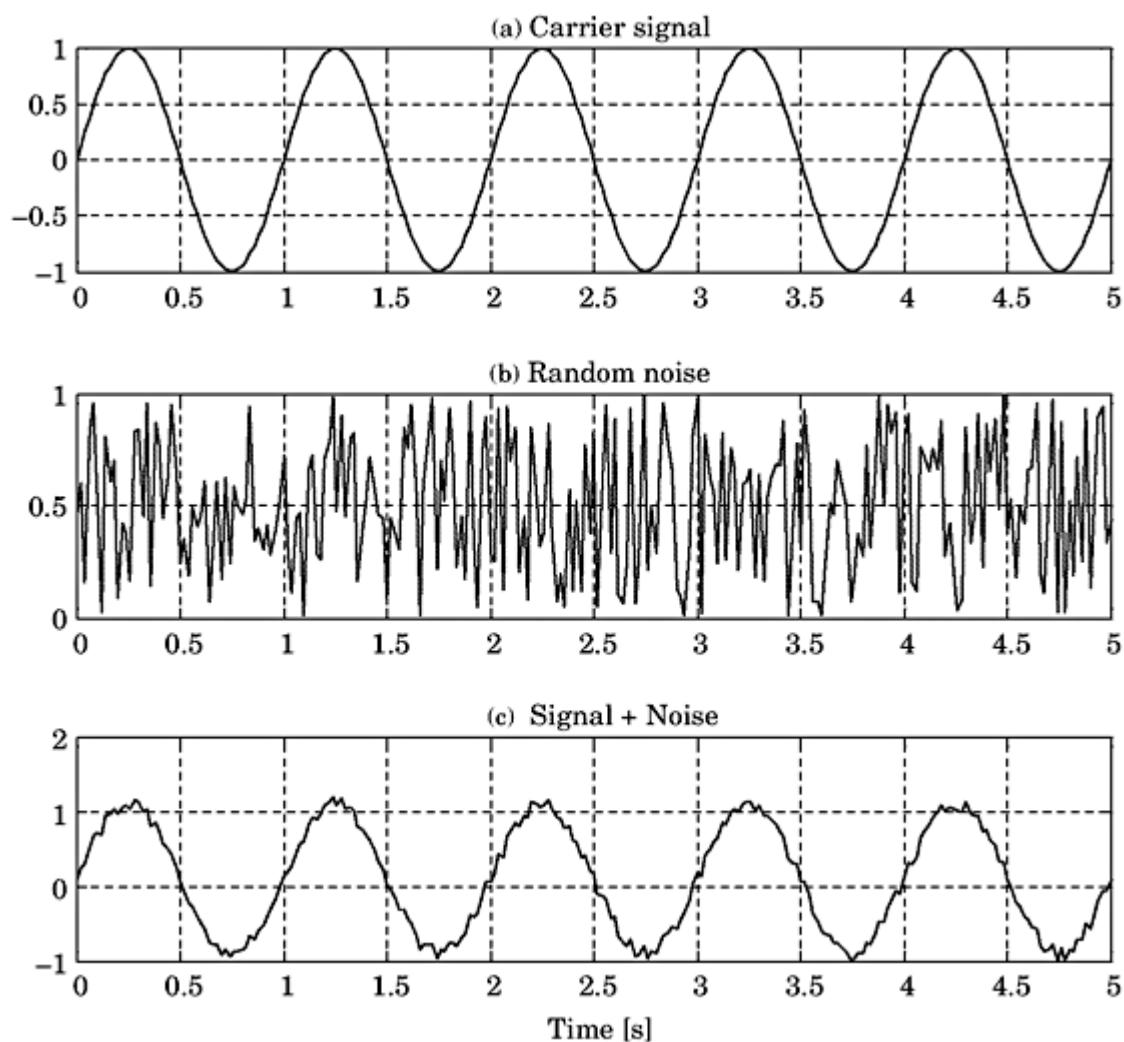
In this chapter, we will deal with the first category of applications, where the PLL must lock onto signals that are corrupted by noise. Before entering into details, we first want to identify the sources and types of noise in PLL systems.

## Sources and Types of Noise in a PLL

In most communications where the PLL is used, digital signals are transmitted—that is, the baseband signals consist of pulses or square wave signals. A binary sequence of 10101010 ..., for example, would be represented by a symmetrical square wave. Due to the harmonics, the bandwidth requirements would become excessive if the pure square-wave signal were sent. To save bandwidth,

only the fundamental is transmitted. The received signal is therefore a sine wave. For an arbitrary sequence of bits, the waveform gets more complex of course, but the transitions still look like “sine waves,” and the received signal must be considered “analog.”

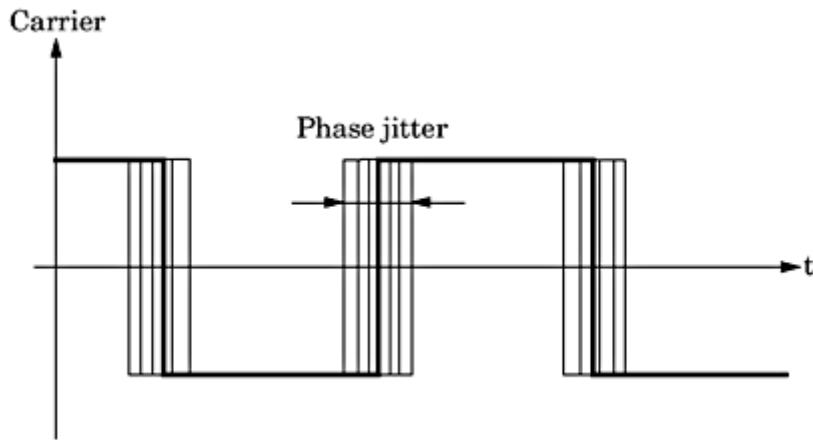
In every communication link, the transmitted signal picks up noise. Noise can be generated in every amplifier and in repeaters, but it also can be created by crosstalk from other channels, from atmospheric noise and many other sources. However it is generated, all that noise is added algebraically to the (analog) data signal. This is sketched by Fig. 4.1 for the trivial case where the data signal is a bit stream of the form 10101010 ... Many types of additive noise exist. The most common is called AWGN (*added white Gaussian noise*). The term “Gaussian” characterizes the amplitude distribution of the signal, saying that the probability density function of the noise amplitude is normally distributed (a *Gauss* function). The term “white” refers to the spectrum of noise and says that within the so-called noise bandwidth of the channel, every frequency interval  $\Delta f$  contains the same noise power  $dP_n$ —meaning  $dP_n/\Delta f$  is constant. The noise



**Figure 4.1** Added noise to information signal. (a) Shows a very simple signal (a sequence of binary 101010 ...). (b) A random noise signal. (c) The sum of both.

---

**Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.**



**Figure 4.2** When a data signal is reshaped, amplitude noise is converted to phase noise (phase jitter).

shown in [Fig. 4.1](#) is referred to as “amplitude noise” because it modulates the amplitude of the data signal.

In the receiver, a band-limited data signal is usually reshaped, using a Schmitt trigger, for instance. The converted signal is a square wave then. For a binary signal, its amplitude can take only two levels, “high” and “low” respectively. The noise superimposed on the data signal now causes the zero crossings of the square wave to become “jittered,” as shown in [Fig. 4.2](#).

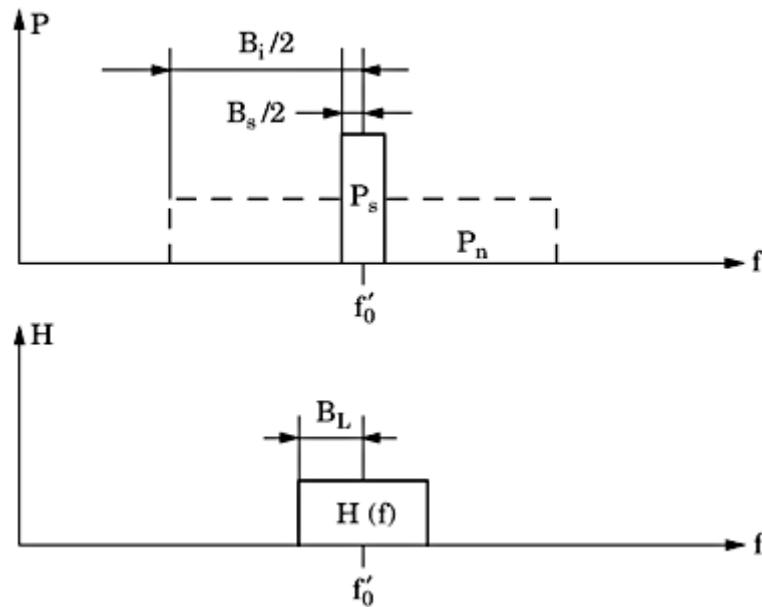
This kind of noise is called *phase noise* or *phase jitter*. The solid curve in [Fig. 4.2](#) is the undistorted data signal, and the thinner lines represent the “jittered” transients. To cope with the effects of superimposed noise, we now define the most important parameters that describe noise.

## Defining Noise Parameters

To analyze the noise performance of the PLL, it is most convenient to describe signals and noise by their power density spectra (see [Fig. 4.3](#)). The upper part shows the spectra of the signal (solid curve, labeled  $P_s$ ) and of noise (dashed curve, labeled  $P_n$ ). The signal spectrum is centered at the down-scaled center frequency  $f_0'$  of the VCO, and its one-sided bandwidth is denoted as  $B_s/2$ . The noise spectrum usually is also symmetrical around the center frequency and is broadband in most cases; its one-sided bandwidth is labeled  $B_n/2$ . Signal power is denoted  $P_s$  in this figure, while noise power is shown as  $P_n$ . The lower part of the figure shows the phase frequency response  $H(f)$  of the PLL. We remember that the phase-transfer function  $H(s)$  relates the Laplace transform  $\Theta_2'(s)$  of the phase  $\theta_2'(t)$  to the Laplace transform  $\Theta_1(s)$  of the phase  $\Theta_1(t)$ . Setting  $s = j\omega$ ,  $H(\omega)$  is the *phase frequency response* of the PLL. As we see from the plot in [Fig. 3.2](#),  $H(\omega)$  is a lowpass filter function. Now the variable  $\omega$  in  $H(\omega)$  is the (radian) frequency of the phase signal that modulates the carrier frequency  $\omega_0'$ , hence  $\omega$  adds to the carrier frequency. If we plot  $H(f)$  versus the sum of carrier + modulation frequency, we get a bandpass function as seen from the lower trace in

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.



**Figure 4.3** Definition of noise parameters, for symbols (cf. text).

**Fig. 4.3.** (Note that  $H$  is plotted versus frequency  $f$  here, and not versus radian frequency  $\omega$ .) This simply means that the PLL is able to track frequencies within a passband centered at the down-scaled center frequency  $f'_0$  ( $f'_0 = \omega'_0 / 2\pi$ ). The symbol  $B_L$  stands for *noise bandwidth* (this term will be defined in [Sec. 4.3](#)). Let us state for the moment that the one-sided noise bandwidth  $B_L/2$  is approximately equal to the earlier-defined 3-dB bandwidth  $f_{3dB}$  ( $f_{3dB} = \omega_{3dB}/2\pi$ ). As we easily recognize from [Fig. 4.3](#), the noise spectrum that is outside of the passband of the PLL will be suppressed by the loop, hence only the part of the noise spectrum within the noise bandwidth is able to corrupt PLL performance. As will be explained in the following sections, two important parameters describe noise performance:

- The signal-to-noise ratio  $SNR_i = P_s/P_n$  at the input of the PLL
- The ratio  $B_i/B_L$  of input noise bandwidth  $B_i$  to PLL noise bandwidth  $B_L$

## The Impact of Noise on PLL Performance

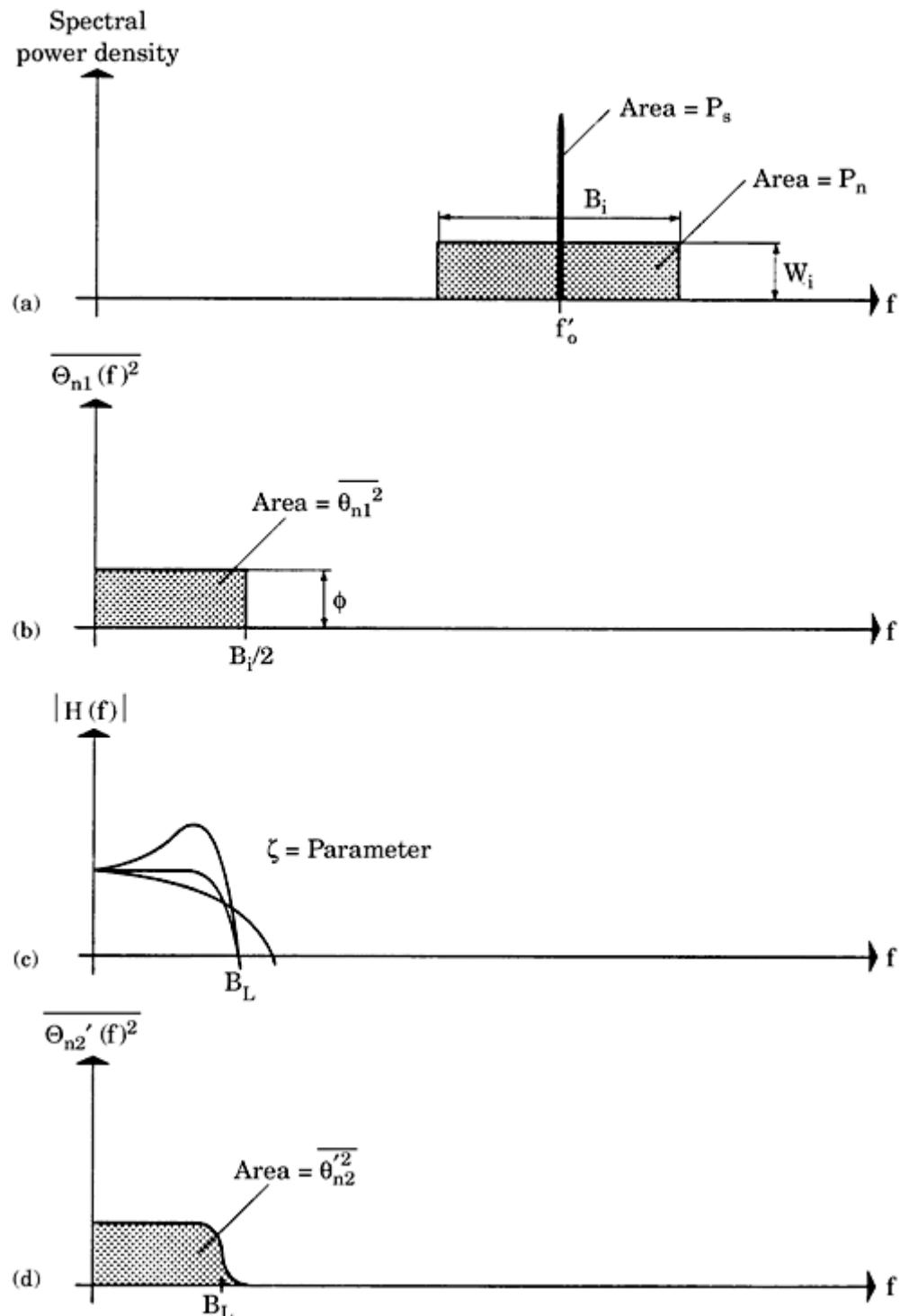
The noise analysis presented here is based on the work of *Gardner*.<sup>1</sup> We shall consider amplitude and power density spectra of information signals and noise signals. Two types of power density spectra have been defined: one-sided and two-sided. If an information signal has one single spectral line at a frequency  $f_0$ —for example, the one-sided power density spectrum shows up a line at  $f_0$ , but the two-sided spectrum would have two spectral lines at  $f_0$  and at  $-f_0$ , respectively. The same applies for noise spectra. It is a matter of taste whether to use one-sided or two-sided power spectra. When using two-sided power spectra, the total power of a signal can be calculated by integrating its power density spectrum over the frequency interval  $-\infty < f < \infty$ . For one-sided spectra, however,

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

this interval becomes  $0 < f < \infty$ , of course. We follow Gardner's terminology in the following and use one-sided spectra only.

Noise analysis is explained by Fig. 4.4, in which input signal and input noise are plotted once again. Here, we assumed for simplicity purposes that the input signal consists of one spectral line at the down-scaled center frequency  $f'_0$ .



**Figure 4.4** Method of calculating the phase jitter  $\overline{\theta_{n2}^2}$  at the down-scaled output of the PLL.

- (a) Power spectra of the reference signal  $u_1(t)$  and the superimposed noise signal  $u_n(t)$ .
- (b) Spectrum of the phase noise at the input of the PLL.
- (c) Bode plot of the phase-transfer function  $H(f)$ .
- (d) Spectrum of the phase noise at the down-scaled output of the PLL.

The noise spectrum has a bandwidth of  $B_i$ . Note that  $B_i$  is finite in all practical cases, because there is always a prefilter or preamplifier in the system that limits noise bandwidth. We now calculate the phase jitter created by the noise at the reference input of the PLL (cf. input  $u_1$  in Fig. 2.1). Assuming the noise spectrum is “white” as explained in the preceding section, Gardner found the mean square value of input phase jitter  $\overline{\theta_{n1}^2}$  to be<sup>1</sup>

$$\overline{\theta_{n1}^2} = \frac{P_n}{2P_s} \quad (4.1)$$

with  $P_s$  = signal power ( $W$ ) and  $P_n$  = noise power ( $W$ ). Note that  $\overline{\theta_{n1}^2}$  is simply the square of the rms value of input phase jitter.

We now define the *SNR* at the input of the PLL as

$$(SNR)_i = \frac{P_s}{P_n} \quad (4.2)$$

For the phase jitter at the input of the PLL, we get the simple relation

$$\overline{\theta_{n1}^2} = \frac{1}{2(SNR)_i} \text{ rad}^2 \quad (4.3)$$

that is, the square of the rms value of the phase jitter is inversely proportional to the *SNR* at the input of the PLL. We remember that  $\overline{\theta_{n1}^2}$  is the mean square of the phase noise that modulates the carrier frequency  $f_0'$  of the PLL ( $f_0' = \omega_0' / 2\pi$ ). Consequently, the spectrum  $\Theta_{n1}(\omega)$  of input phase jitter is identical with the input noise spectrum shifted down by  $f_0'$ .<sup>1</sup> Fig. 4.4b shows the mean square of input phase jitter

$$\overline{\Theta_{n1}^2(\omega)} = \Phi = \frac{\overline{\theta_{n1}^2}}{B_i/2} \text{ rad}^2 \text{ Hz}^{-1} \quad (4.4)$$

Since the noise spectrum has been assumed to be white, the mean square  $\overline{\Theta_{n1}^2(\omega)}$  has the constant value  $\Phi$ . Knowing the spectrum of input phase jitter, we can compute the spectrum  $\overline{\Theta_{n2}^2(\omega)}$  of the down-scaled output phase jitter from

$$\overline{\Theta_{n2}^2(\omega)} = |H(\omega)|^2 \cdot \overline{\Theta_{n1}^2(\omega)} = |H(\omega)|^2 \Phi \quad (4.5)$$

Fig. 4.4c plots the Bode diagram of  $H(\omega)$ , and Fig. 4.4d shows the mean square  $\overline{\Theta_{n2}^2(\omega)}$  of the output phase noise spectrum, which is simply the multiplication of curves (b) and (c) in the figure. To compute the mean square of the output phase jitter  $\overline{\theta_{n2}^2}$ , we use Parseval’s theorem to get

$$\overline{\theta'^2_{n2}} = \int_0^{\infty} \overline{\Theta'^2_{n2}}(2\pi f) df \quad (4.6)$$

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

where  $2\pi f = \omega$ .  $\overline{\theta_{n2}^{'2}}$  is the area under the curve in Fig. 4.4d. Making use of Eqs. (4.5) and (4.6), we get

$$\overline{\theta_{n2}^{'2}} = \int_0^\infty \Phi |H(2\pi f)|^2 df = \frac{\Phi}{2\pi} \int_0^\infty |H(\omega)|^2 d\omega \quad (4.7)$$

The integral  $\int_0^\infty |H(2\pi f)|^2 df$  is called noise bandwidth  $B_L$ ,

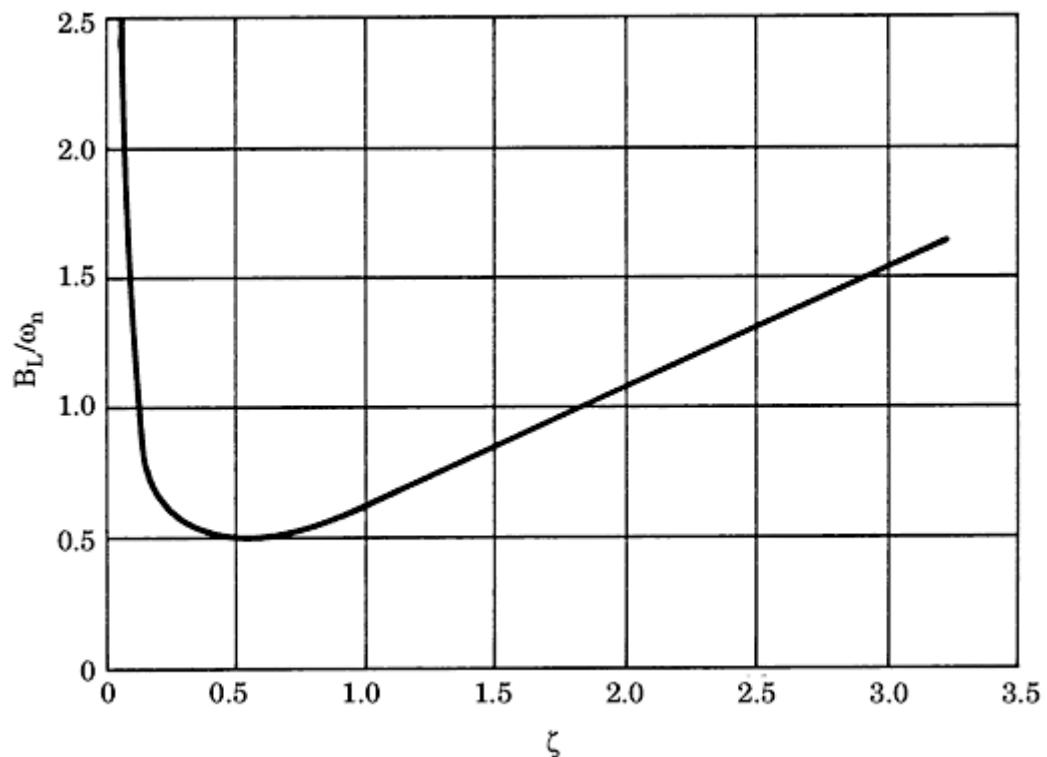
$$B_L = \int_0^\infty |H(2\pi f)|^2 df \quad (4.8)$$

The solution of this integral reads<sup>1</sup>

$$B_L = \frac{\omega_n}{2} \left( \zeta + \frac{1}{4\zeta} \right) \quad (4.9)$$

Thus,  $B_L$  is proportional to the natural frequency  $\omega_n$  of the PLL; furthermore, it depends on the damping factor  $\zeta$ . Fig. 4.5 is a plot of  $B_L$  versus  $\zeta$ , and  $B_L$  has a minimum at  $\zeta = 0.5$ . In this case, we have

$$B_L = B_{L\min} = \frac{\omega_n}{2} \quad (4.10)$$



**Figure 4.5** The noise bandwidth of a second-order PLL as a function of the damping factor  $\zeta$   
(Adapted from Gardner<sup>1</sup> with permission).

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

In [Sec. 3.4](#), we showed that the transient response of the PLL is best at  $\zeta = 0.7$ . Because the function  $B_L(\zeta)$  is fairly flat in the neighborhood of  $\zeta = 0.5$ , the choice of  $\zeta = 0.7$  does not noticeably worsen the noise performance. For  $\zeta = 0.7$ ,  $B_L$  is  $0.53 \omega_n$  instead of the minimum value  $0.5 \omega_n$ . For this reason,  $\zeta = 0.7$  is chosen for most applications.

For the output phase jitter  $\overline{\theta_{n2}^{\prime 2}}$ , we can now write

$$\overline{\theta_{n2}^{\prime 2}} = \Phi B_L \quad (4.11)$$

where  $\Phi$  is already known from [Eq. \(4.4\)](#). Combining Eqs. [\(4.1\)](#), [\(4.3\)](#), and [\(4.10\)](#), we obtain

$$\overline{\theta_{n2}^{\prime 2}} = \frac{P_n}{P_s} \cdot \frac{B_L}{B_i} \quad (4.12)$$

We saw in [Eq. \(4.3\)](#) that the phase jitter at the input of the PLL is inversely proportional to the  $(SNR)_i$ . By analogy, we can also define a signal-to-noise ratio at the output, which will be denoted by  $(SNR)_L$  ( $SNR$  of the loop). We define this analogy in [Eq. \(4.13\)](#):

$$\overline{\theta_{n2}^{\prime 2}} = \frac{1}{2(SNR)_L} \quad (4.13)$$

Comparing Eqs. [\(4.12\)](#) and [\(4.13\)](#), we get

$$(SNR)_L = \frac{P_s}{P_n} \cdot \frac{B_i}{2B_L} = (SNR)_i \cdot \frac{B_i}{2B_L} \quad (4.14)$$

[Equation \(4.14\)](#) says that the PLL improves the  $SNR$  of the input signal by a factor of  $B_i/2B_L$ . The narrower the noise bandwidth  $B_L$  of the PLL, the greater the improvement.

All this sounds very theoretical. Let us therefore look at some numerical data. In radio and television, the  $SNR$  is used to specify the quality of information transmission. For a stereo receiver, a minimum  $SNR$  of 20 dB is considered a fair design goal. The same holds true for PLLs. Practical experiments performed with second-order PLLs have demonstrated some very useful results.<sup>1</sup>

1. For  $(SNR)_L = 1$  (0 dB), a lock-in process will not occur because the output phase noise  $\overline{\theta_{n2}^{\prime 2}}$  is excessive.
2. At  $(SNR)_L = 2$  (3 dB), lock-in is eventually possible.
3. For  $(SNR)_L = 4$  (6 dB), stable operation is generally possible.

In quantitative terms according to [Eq. \(4.13\)](#), for  $(SNR)_L = 4$  the output phase noise becomes

$$\overline{\theta_{n2}^{\prime 2}} = \frac{1}{2 \cdot 4} = \frac{1}{8} \text{ rad}^2$$

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

Hence, the rms value  $\sqrt{\theta'^2_2}$  becomes

$$\sqrt{\theta'^2_2} = \sqrt{1/8} = 0.353 \text{ rad}$$

Since the rms value of phase jitter is about  $20^\circ$ , the value of  $180^\circ$  (limit of dynamic stability) is rarely exceeded. Consequently, the PLL does not unlock frequently. At  $(SNR)_L = 1$ , however, the rms value of output phase jitter would be as large as  $40^\circ$ , and the dynamic limit of stability would be exceeded on every major noise peak, thus making stable operation impossible.

As a rule of thumb,

$$(SNR)_L \geq 4 (\approx 6 \text{ dB}) \quad (4.15)$$

is a convenient design goal.

**Note:** The SNR of a signal can be specified either numerically or in decibels. The numerical value SNR is computed from

$$SNR = \frac{P_s}{P_n} = \frac{U_s^2(\text{rms})}{U_n^2(\text{rms})}$$

whereas the  $(SNR)_{dB}$  is calculated from

$$(SNR)_{dB} = 10\log_{10} \frac{P_s}{P_n} = 20\log_{10} \frac{U_s(\text{rms})}{U_n(\text{rms})}$$

where  $U_s$  (rms) and  $U_n$  (rms) are the rms values of signal and noise, respectively.

The designer of practical PLL circuits is vitally interested in how often, on the average, a system will temporarily unlock. The probability of unlocking is decreased with increasing  $(SNR)_L$ . We now define  $T_{av}$  to be the average time interval between two lock-outs. For example, if  $T_{av} = 100$  ms, the PLL unlocks (on average) 10 times per second.

For second-order PLLs,  $T_{av}$  has been found experimentally as a function of  $(SNR)_L$ .<sup>1</sup> The resulting curve is plotted in Fig. 4.6.

To illustrate the theory, let's calculate a numerical example.

**Numerical Example** A second-order PLL is assumed to have the following specifications:

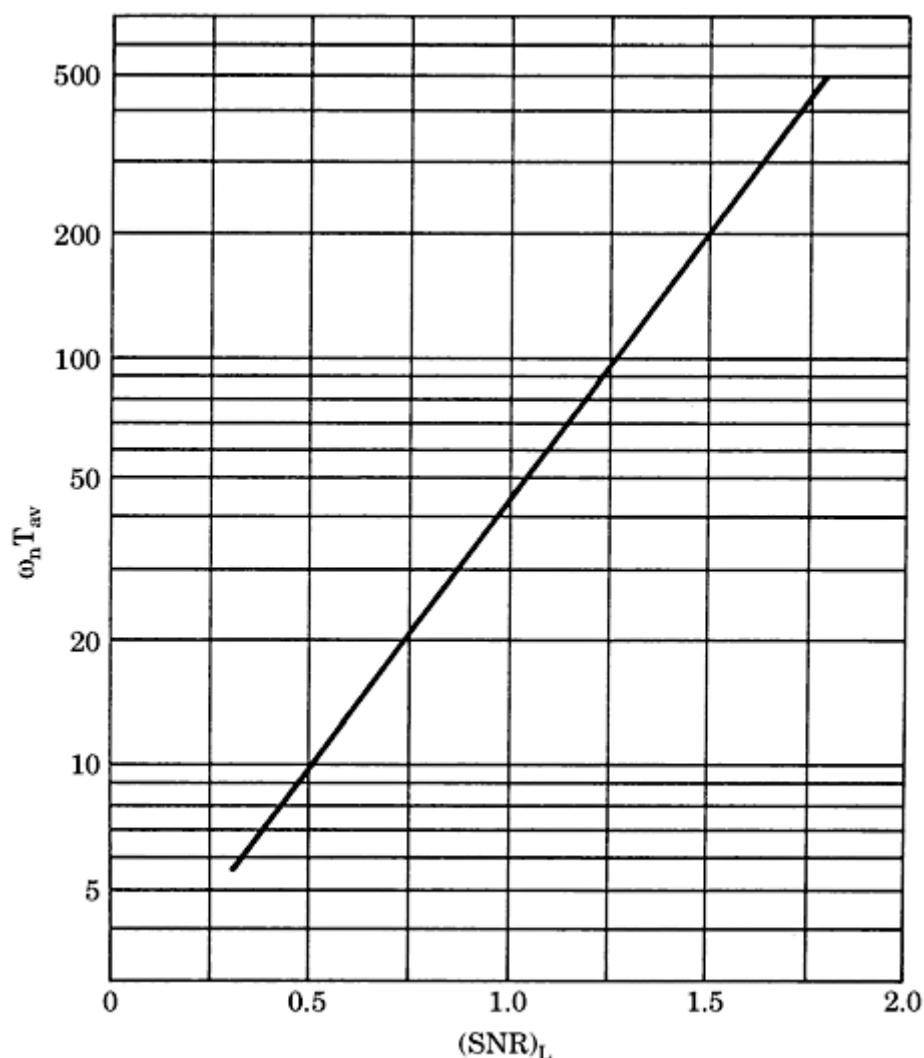
$$f_n = 10 \text{ kHz}$$

$$\omega_n = 62.8 \cdot 10^3 \text{ s}^{-1}$$

$$(SNR)_L = 1.5$$

From Fig. 4.6, we read  $\omega_n T_{av} = 200$ . Consequently,  $T_{av} = 3 \text{ ms}$ . This means the PLL unlocks

about 300 times per second, which looks quite bad. However, the lock-in time  $T_L$  is found from Eq. (3.62) to be  $T_L \approx \frac{2\pi}{\omega_n} = 100 \mu s$ —in other words, the PLL is locked 97 percent of the total time.



**Figure 4.6** Average time  $T_{av}$  between two succeeding unlocking events plotted as a function of the  $(SNR)_L$  at the output of the PLL.  $T_{av}$  is normalized to the natural frequency  $\omega_n$  of the PLL. Note that this result is valid only for linear second-order PLLs. (Adapted from Gardner<sup>L</sup> with permission.)

**Summary of noise theory.** Although the noise theory of the PLL is difficult, for practical design purposes it's important to remember some rules of thumb:

- Stable operation of the PLL is possible if  $(SNR)_L$  is approximately 4.
- $(SNR)_L$  is calculated from

$$(SNR)_L = \frac{P_s}{P_n} \cdot \frac{B_i}{2B_L}$$

where  $P_s$  = signal power at the reference input

$P_n$  = noise power at the reference input

$B_i$  = bandwidth of the prefilter, if any; if no prefilter is used,

$B_i$  is the bandwidth of the signal source (for example,  
antenna, repeater)

$B_L$  = noise bandwidth of the PLL

- The noise bandwidth  $B_L$  is a function of  $\omega_n$  and  $\zeta$ . For  $\zeta = 0.7$ ,  $B_L = 0.53 \omega_n$ .
- The average time interval  $T_{av}$  between two unlocking events gets longer as  $(SNR)_L$  increases.

**Some critical remarks on noise theory.** The noise theory presented in this section is based on the assumption that signal and noise at the reference input of the PLL are stationary. The term *stationary* means the statistical parameters of signal and noise do not change with time—that is, the power spectrum of noise always remains the same. In many applications of communication, this condition is not met: the reference signal can almost disappear in some time intervals due to fading, for example. When the input signal of a PLL gets momentarily lost, performance of the PLL depends largely on the type of phase detector used. As we will see, the loop filter also has some influence, though minor.

When comparing the four types of phase detectors discussed in this chapter, we note that the multiplier and EXOR phase detectors are level-sensitive, whereas the JK-flipflop and PFD are edge-sensitive. Assume the PLL uses the multiplier phase detector. If the input signal  $u_1$  fades away, it becomes nearly zero, and only noise is left. Because that noise is uncorrelated with the (down-scaled) output signal  $u_2'$  of the VCO (cf. Fig. 2.1), the average output signal  $\overline{u_d}$  is zero. The situation is about the same when the EXOR is utilized. When the reference signal  $u_1$  disappears, the signal at the reference input (after reshaping) hangs up at either a “low” or a “high” level. Consequently, the output signal of the EXOR is either identical with the down-scaled VCO output signal  $u_2'$  or with the logically inverted signal  $u_2'$ . The average value  $\overline{u_d}$  is zero again. What happens now with the output signal of the loop filter? The situation is similar for the passive and the active lead-lag filter. With a zero input, the output is also pulled toward zero with a time constant  $\tau_1 + \tau_2$  for the passive, or with a time constant  $\tau_1$  for the active lead-lag filter. Hence, the frequency created by the VCO will relatively slowly drift away toward the center frequency  $\omega_0$ . When the reference signal disappears for an extended period of time, the loop surely will get out of lock. If the active PI filter was used, then the capacitor  $C_1$  (cf. Fig. 2.21a) has been charged to that voltage that was required to create the appropriate VCO frequency before the reference input faded away. With zero input signal, the voltage across capacitor  $C_1$  remains nearly unchanged, thus the instantaneous output frequency of the VCO is held nearly constant. There is a good chance the loop is still in lock when the reference signal reappears.

With edge-sensitive phase detector types, the situation is worse. In case of the JK-flipflop, the phase detector output hangs up at a logic “low” level after the reference signal disappears. Consequently, the output signal of the loop filter will be quickly pulled down to its minimum level, and the VCO output frequency also runs away quickly. The same holds true for the PFD. As soon as the reference signal fades away, the PFD will go into its  $-1$  state (output at logic “low”). Again the VCO output frequency will run away quickly.

We conclude that in cases where the reference signal can drop out, phase detector types 1 or 2 are the better choice. Moreover, the active PI loop filter (type 3) offers the best performance relating to run-away of the VCO output frequency.

## Pull-in Techniques for Noisy Signals

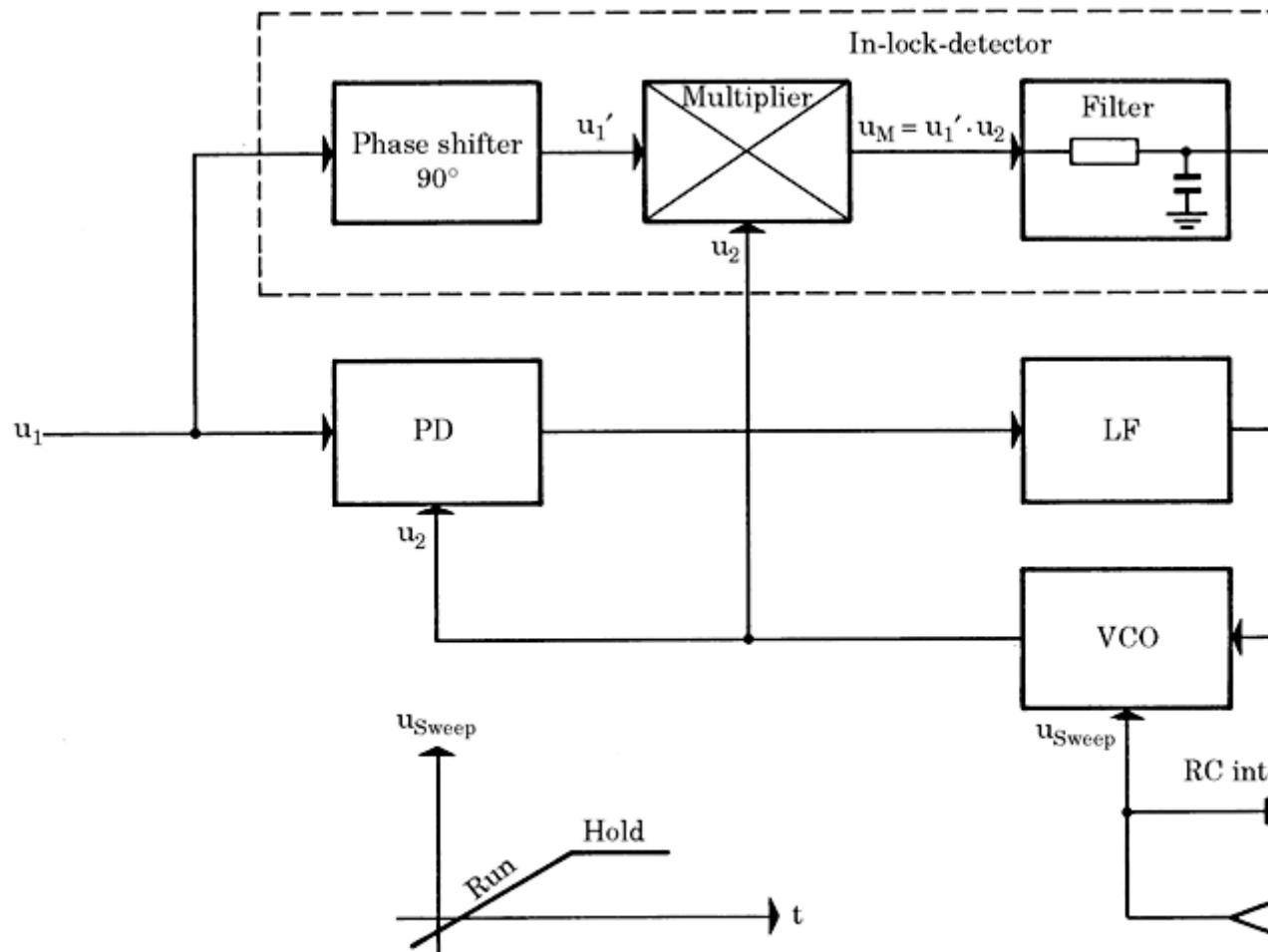
In this section, we summarize the most popular pull-in techniques. A special pull-in procedure becomes mandatory if the noise bandwidth of a PLL system must be made so narrow the signal may possibly not be captured at all.

### The sweep technique

If this procedure is chosen, the noise bandwidth  $B_L$  is made so small that the  $SNR$  of the loop ( $SNR$ )<sub>L</sub> is sufficiently large to provide stable operation. As a consequence, the lock range  $\Delta\omega_L$  might become smaller than the frequency interval  $\Delta\omega$  within which the input signal is expected to be. To solve the locking problem, the center frequency of the VCO is swept by means of a sweeping signal  $u_{\text{sweep}}$  (Fig. 4.7) over the frequency range of interest. Of course, the sweep rate must be held within the limits specified by Eq. (3.40); otherwise, the PLL could not become locked. (In Sec. 3.4, we considered the case where the center frequency  $\omega_0$  was constant and the reference frequency  $\omega_1$  was swept. In the case considered here, the reference frequency is assumed to be constant, whereas the center frequency is swept. For the PLL, both situations are equivalent, because the frequency offset  $\Delta\omega = \omega_1 - \Delta\omega_0'$  is the only parameter of importance.)

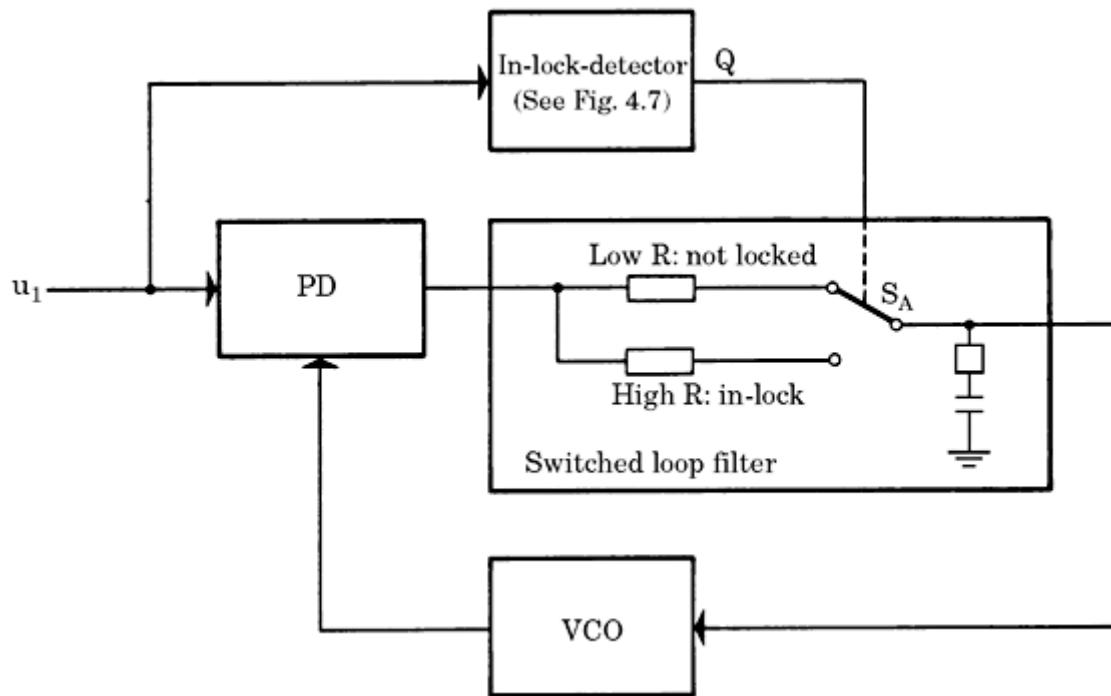
As shown in the block diagram of Fig. 4.7, the center frequency of the VCO can be tuned by the signal applied to its sweep input. A linear sawtooth signal generated by a simple  $RC$  integrator is used as the sweep signal. Assume the PLL has not yet locked. The integrator is in its RUN mode, and hence the sweep signal builds up in the positive direction. As soon as the frequency of the VCO approaches the frequency of the input signal, the PLL suddenly locks. The sweep signal should now be frozen at its present value (otherwise, the VCO frequency would run away). This is realized by throwing the analog switch in Fig. 4.7 to the HOLD position. To control the analog switch, we need a signal that tells us whether or not the PLL is in the locked state. Such a control signal is generated by the in-lock detector shown in Fig. 4.7. In this example, the in-lock detector is a cascade connection of a  $90^\circ$  phase shifter, an analog multiplier, a low-pass filter, and a Schmitt trigger. If the PLL is locked, there is a phase offset of approximately  $90^\circ$  between input signal  $u_1$  and VCO output signal  $u_2$ . (Note: it was assumed here that the phase detector is either type 1 or 2. For other phase detectors, the phase relationship would be different.) The phase shifter then outputs a signal  $u_1'$ , which is nearly in phase with the VCO output signal  $u_2$ . The average value of the output signal of the multiplier  $u_M = u_1' \cdot u_2$  is positive. If the PLL is not in the locked state, however, the signals  $u_1'$  and  $u_2$  are uncorrelated, and the average value of  $u_M$  is zero. Thus, the output signal of the multiplier is a clear indication of lock. To eliminate AC components and

**Any use is subject to the Terms of Use as given at the website.**



**Figure 4.7** A simplified block diagram of a PLL using the sweep technique for the acquisition of signal

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.



**Figure 4.8** A simplified block diagram of a PLL using a switched loop filter for the acquisition of noisy signals. ( $S_A$  = analog switch.)

to inhibit false triggering, the  $u_M$  signal is conditioned by a low-pass filter. The filtered signal is applied to the input of the Schmitt trigger.

If the PLL has locked, the integrator is kept in the HOLD mode, as mentioned. Of course, an analog integrator would drift away after some time. To avoid this effect, an antidrift circuit must be added—however, this is not shown in Fig. 4.7.

### The switched-filter technique

This locking method is depicted in Fig. 4.8. This configuration uses a loop filter whose bandwidth can be switched by a binary signal. The control signal for the switched filter is also derived from an in-lock detector, as shown in the previous example. In the unlocked state of the PLL, the output signal  $Q$  of the in-lock detector is zero. In this state, the bandwidth of the loop filter is so large the lock range exceeds the frequency range within which the input signal is expected. The noise bandwidth is then too large to enable stable operation of the loop. There is nevertheless a high probability that the PLL will lock spontaneously at some time. To avoid repeated unlocking of the loop, the filter bandwidth has to be reduced instantaneously to a value where the noise bandwidth  $B_L$  is small enough to provide stable operation. This is done by switching the loop filter to its low-bandwidth position by means of the  $Q$  signal.

## Design Procedure for Mixed-Signal PLLs

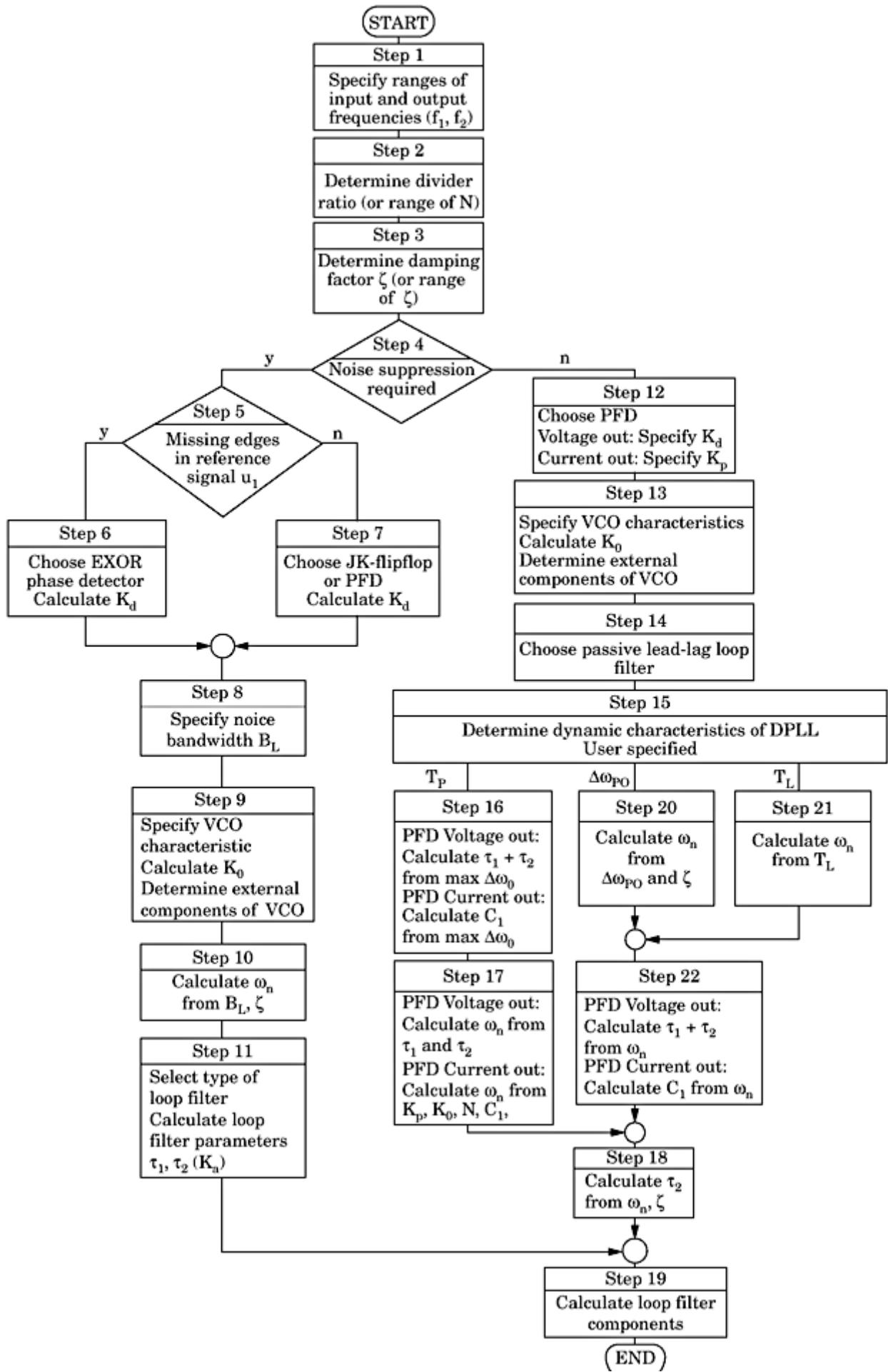
The mixed-signal PLL can be built in many variants, and the spectrum of applications is very broad as well. There are PLL applications in communications where the system is used to extract the clock from a—possibly noisy—information signal. In such an application, noise suppression is of importance. An entirely different application of the PLL is frequency synthesis. Here, reference noise is not of concern, but the synthesizer should be able to switch rapidly from one frequency to another; hence, pull-in time is the most relevant parameter.

For these reasons it appears difficult to give a design procedure that yields an optimum solution for every PLL system. In this section, we present a PLL design procedure that is based on a flowchart (see [Fig. 5.1](#)). To help compute the relevant entities in the PLL design, the often-used formulas for the PLL key parameters have been listed in five tables in [chapter 3](#), Tables [3.1](#) through [3.5](#). [Table 3.1](#) shows the equations for hold range, lock range, and so on for a PLL using a type-1 phase detector. [Table 3.2](#) presents the same information, but for the EXOR phase detector, and so on.

The step-by-step design procedure of [Fig. 5.1](#) should not be considered a universal tool for the thousand-and-one uses of the PLL, but rather as a series of design hints. Moreover, in many cases the design of a PLL will be an iterative process. We may start with some initial assumptions but end up perhaps with a design that is not acceptable, because one or more key parameters (for example, pull-in time) are outside the planned range. In such a situation, we restart with altered premises and repeat the procedure until the final design appears acceptable.

Manufacturers of PLL ICs have already provided design tools running on the PC. A program distributed by Philips,[52](#) for example, is used to design PLL systems using the popular integrated circuits 74HC/HCT4046A, 74HC/HCT7046A, and 74HCT9046A. All three are based on the old industry standard CD4046 IC (from the 4000 CMOS series), which was originally introduced by RCA.





**Figure 5.1** Flowchart of the mixed-signal PLL design procedure.

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

The design procedure presented here is independent of the type of PLL circuit that will finally be used to implement the system. The individual steps are described in the following. As mentioned, most of the formulas used to design the PLL are listed in Tables 3.1 through 3.5. In the procedure explained in the following, the VCO is assumed to be a relaxation oscillator VCO (as shown in Fig. 2.23). With some slight modifications, however, it can also be applied to resonant-type VCOs. We add some comments at the end of this chapter regarding the design of resonant VCOs that use varactor diodes to vary the output frequency.

**Step 1.** In the first step, the input and output frequencies of the PLL must be specified. There are cases where both input frequency and output frequency are constant but not necessarily identical. In other applications (for example, frequency synthesizers), the input frequency is always the same, but the output frequency is variable. As a last variant, both input and output frequencies could be variable. Let  $f_{1\min}$  and  $f_{1\max}$  be the minimum and maximum input frequencies, and  $f_{2\min}$  and  $f_{2\max}$  the minimum and maximum output frequencies, respectively.

**Step 2.** In this step, the scalar ratio must be determined. In some PLL applications, the output frequency  $f_2$  always equals the reference frequency  $f_1$ . Here no down-scaler is needed—in other words,  $N = 1$ . Cases exist where the ratio of output to reference frequency is greater than 1 but remains fixed. Here, a down scaler with constant divider ratio  $N$  is required. When the PLL is used to build a frequency synthesizer, the ratio of output to reference frequency is variable; thus, a range for  $N$  must be defined ( $N_{\min} \leq N \leq N_{\max}$ ). When  $N$  is variable, natural frequency  $\omega_n$  and damping factor  $\zeta$  will vary with  $N$ , as seen from the corresponding equations in Tables 3.1 through 3.5. Both of these parameters will vary approximately with  $1/\sqrt{N}$ . Consequently,  $\omega_n$  will vary in the range  $\omega_{n\min} < \omega_n < \omega_{n\max}$ , and  $\zeta$  will vary in the range  $\zeta_{\min} < \zeta < \zeta_{\max}$ . For these ranges, we get approximately

$$\frac{\omega_{n\max}}{\omega_{n\min}} = \sqrt{\frac{N_{\max}}{N_{\min}}} \quad (5.1)$$

and

$$\frac{\zeta_{\max}}{\zeta_{\min}} = \sqrt{\frac{N_{\max}}{N_{\min}}} \quad (5.2)$$

respectively. As we know, a damping factor between 0.5 and 1 is considered optimum. As long as the ratio  $N_{\max}/N_{\min}$  is not too large, the variation of the damping factor can be accepted; if  $N$  varies by a factor of 10, for example,  $\zeta$  varies by about a factor of 3, which can be tolerated.

Much larger variations of  $\zeta$ , however, must be avoided, because the loop then would get oscillatory for the smallest, and become sluggish for the largest damping factor. When  $N$  varies over a large range (for instance, 1 to 100), it is often

**Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.**

mandatory to define more than one frequency range for the PLL and switch the range accordingly. To ease the design in the case of variable  $N$ , we specify the parameters of the PLL such that  $\zeta$  becomes optimum for a divider ratio  $N_{\text{mean}}$ , which is given by the geometric mean of  $N_{\text{max}}$  and  $N_{\text{min}}$ ,

$$N_{\text{mean}} = \sqrt{N_{\text{min}} N_{\text{max}}} \quad (5.3)$$

(For constant  $N$ ,  $N_{\text{mean}} = N$ , of course.) If  $N_{\text{min}} = 10$  and  $N_{\text{max}} = 100$ , for example,  $N_{\text{mean}}$  would be  $31.6 \rightarrow 32$ . Choosing  $\zeta = 0.7$  for  $N = 32$  would yield a minimum damping factor of  $\zeta_{\text{min}} = 0.4$  and a maximum of  $\zeta_{\text{max}} = 1.2$ , which is a fair compromise.

**Step 3.** Determination of damping factor  $\zeta$ . When  $N$  is constant,  $\zeta$  remains constant, too, and can be chosen arbitrarily. For constant  $N$ , it is optimum to select  $\zeta = 0.7$ ; the PLL then has a Butterworth response, as explained in [Sec. 3.3.1](#). If  $N$  is variable, however, it is recommended to choose  $\zeta = 0.7$  for  $N = N_{\text{mean}}$ , as explained in step 2.

**Step 4.** In this step, the question must be answered as to whether the PLL should offer noise suppression or not. If a digital frequency synthesizer must be built, for example, noise can be discarded, and parameters such as noise bandwidth  $B_L$  must not be considered. If noise must be suppressed, however,  $B_L$  and related parameters must be taken into account. If noise is of concern, the procedure continues at step 5, otherwise at step 12.

**Step 5.** Noise must be suppressed by this PLL. As discussed in [Sec. 4.3](#), the various digital phase detectors behave differently in the presence of noise. In a situation where edges of the reference (input) signal  $u_1$  can get lost, edge-sensitive phase detectors such as the JK-flipflop or the PFD then can hang up in one particular state: the output of the JK-flipflop will switch into the “low” state, and the output of the PFD will switch to the state  $-1$  after a very short time. Consequently, the frequency of the VCO will run away quickly, which is certainly undesirable. The average output signal  $u_d$  of the EXOR phase detector, however, will stay at 0 when edges of  $u_1$  are missing. Another option would be the multiplier phase detector, because this detector is also level-sensitive and performs similarly to the EXOR. If edges of  $u_1$  are likely to get lost, the procedure continues at step 6, otherwise at step 7.

**Step 6.** The EXOR phase detector (or the multiplier) should be selected. In case of the multiplier PD, the detector gain must be taken from the data sheet of the corresponding device. When the EXOR is chosen and runs from a unipolar power supply,  $K_d$  is given by  $K_d = U_B/\pi$ , where  $U_B$  is the supply voltage. When a bipolar power supply is used, or when the EXOR saturates at levels that differ substantially from the power-supply rails, use [Eq. \(2.19\)](#). The procedure continues with step 8.

**Step 7.** The JK-flipflop or the PFD can be chosen for the phase detector. As explained in [Sec. 2.4.4](#), the PFD offers superior performance—for example, infinite

pull-in range—so this type of phase detector is normally preferred. When a voltage output PFD is used, the phase detector gain  $K_d$  can now be determined. When a unipolar power supply is used,  $K_d = U_B/2\pi$  for the JK-flipflop or  $K_d = U_B/4\pi$  for the PFD ( $U_B$  = supply voltage). When a bipolar power supply is used, or when the phase detector saturates at voltage levels that differ substantially from the power-supply rails, the corresponding equation given in [Sec. 2.4](#) should be used. When a current output PFD is employed, the detector gain  $K_P$  must be determined. This specification is normally found on the data sheet of the corresponding IC. The procedure continues at step 8.

**Step 8.** The noise bandwidth  $B_L$  must now be specified. As shown in [Sec. 4.3](#),  $B_L$  is related to the signal-to-noise ratio of the loop  $(\text{SNR})_L$  by

$$(\text{SNR})_L = (\text{SNR})_i \frac{B_i}{2B_L} \quad (5.4)$$

$B_L$  should be chosen such that  $(\text{SNR})_L$  becomes larger than some minimum value, typically larger than 4 (which corresponds to 6 dB). If  $(\text{SNR})_i$  is not known, it must be estimated or eventually measured.<sup>17</sup> Finally, the noise bandwidth  $B_i$  at the input of the PLL must also be known.  $B_i$  is nothing else than the bandwidth of the signal source or the bandwidth of an optional prefilter. After  $(\text{SNR})_i$  and  $B_i$  are determined,  $B_L$  can be calculated from [Eq. \(5.4\)](#).

An additional problem arises when the scalar ratio  $N$  must be variable. We know from noise theory that  $B_L$  is also related to  $\omega_n$  and  $\zeta$  by [Eq. \(4.9\)](#), which reads

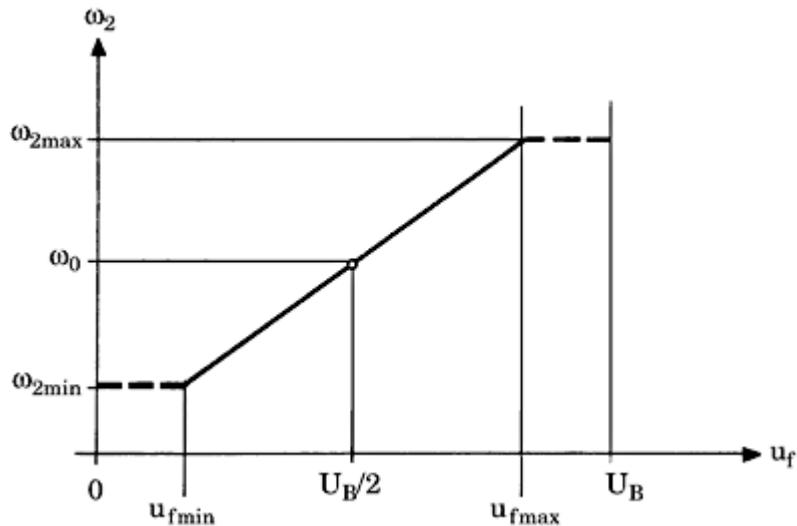
$$B_L = \frac{\omega_n}{2} \left( \zeta + \frac{1}{4\zeta} \right) \quad (5.5)$$

When the divider ratio  $N$  is variable, both  $\omega_n$  and  $\zeta$  vary with  $N$ ; hence,  $B_L$  also becomes dependent on  $N$ . In such a case, we specify  $B_L$  for the case  $N = N_{\text{mean}}$ , as pointed out in step 2. To make sure  $B_L$  does not fall below the minimum acceptable value, we should check its values at the extremes of the scalar ratio  $N$ , using [Eq. \(5.5\)](#). If  $B_L$  becomes too small at one of the extremes of  $N$ , the initial value assumed for  $N = N_{\text{mean}}$  should be increased correspondingly.

**Step 9.** In this step, the characteristic of the VCO will be determined. As mentioned earlier, the procedure is restricted to relaxation VCOs. When a resonant VCO is used, however, the design can be adapted correspondingly; refer to the comments at the end of this chapter.

Because the center frequency  $\omega_0$  (or the range of  $\omega_0$ ) is known and the range of the divider ratio  $N$  is also given, we can calculate the range of output (angular) frequencies that must be generated by the VCO. Let  $\omega_{2\min}$  and  $\omega_{2\max}$  be the minimum and maximum output frequencies of the VCO, respectively. A suitable VCO must be selected first. In most cases, the VCO will be part of the PLL system, typically an integrated circuit. Given the supply voltage(s) of the VCO, the data sheet indicates the usable range of control voltage  $u_f$ . (For a

unipolar



**Figure 5.2** The characteristic of the VCO. The curve shows angular frequency  $\omega_2$  versus control voltage  $u_f$ . Note that the useful voltage range is less than the supply voltage.

power supply with  $U_B = 5$  V, this range is typically 1 to 4 V.) Let  $u_{f\text{min}}$  and  $u_{f\text{max}}$  be the lower and upper limit of that range, respectively. The VCO is now designed such that it generates the output frequency  $\omega_2 = \omega_{2\text{min}}$  for  $u_f = u_{f\text{min}}$ , and the output frequency  $\omega_2 = \omega_{2\text{max}}$  for  $u_f = u_{f\text{max}}$ . The corresponding VCO characteristic is plotted in Fig. 5.2. Now the VCO gain  $K_0$  is calculated from the slope of this curve—in other words:

$$K_0 = \frac{\omega_{2\text{max}} - \omega_{2\text{min}}}{u_{f\text{max}} - u_{f\text{min}}}$$

Now the external components of the VCO can be determined also. Usually the data sheets tell you how to calculate the values of these elements.

**Step 10.** Calculation of natural frequency  $\omega_n$ . Given noise bandwidth  $B_L$  and damping factor  $\zeta$ ,  $\omega_n$  can be calculated from

$$B_L = \frac{\omega_n}{2} \left( \zeta + \frac{1}{4\zeta} \right)$$

When the divider ratio  $N$  is variable, both  $B_L$  and  $\zeta$  have been determined for  $N = N_{\text{mean}}$ ; thus, the value obtained for  $\omega_n$  is valid only for  $N = N_{\text{mean}}$  (see the notes in step 2).

**Step 11.** Specification of the loop filter. The appropriate type of loop filter must be chosen first. The passive lead-lag loop filter is the simplest. If it is combined with a multiplier, EXOR, or JK-flipflop phase detector, however, the pull-in range becomes limited. If “infinite pull-in range” is desired, the active PI loop filter should be selected. For all cases where a current-output PFD is not applied,

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

the equations for  $\omega_n$  and  $\zeta$  in Tables 3.1 through 3.5 are used to calculate the two time constants  $\tau_1$  and  $\tau_2$  from the known values of  $\omega_n$ ,  $\zeta$ ,  $K_0$ ,  $K_d$ , and  $N$ . When the active lead-lag filter is used, a suitable value for  $K_a$  must be chosen in addition. Only values of  $K_a > 1$  are reasonable, of course. Typically,  $K_a$  is in the range 2 through 10. When a current output PFD is used, however, time constant  $\tau_2$  and the value of capacitor  $C_1$  (cf. Fig. 2.17b) must be computed using the equations for  $\omega_n$  and  $\zeta$  in Table 3.5 from the known values of  $\omega_n$ ,  $\zeta$ ,  $K_0$ ,  $K_P$ , and  $N$ . The design proceeds with step 19.

**Step 12.** (Continued from step 4.) Because noise at the reference input can be discarded, the best choice for the phase detector is the PFD. We have the choice between a voltage output and a current output PFD. When the voltage output PFD is applied and runs from a unipolar power supply with supply voltage  $U_B$ , its detector gain  $K_d$  is given by  $K_d = U_B/4\pi$ . When a bipolar power supply is used or when the logic levels differ substantially from the voltages of the supply rails, use the general formula given in Sec. 2.4.4.1. In case of the current-output PFD, the detector gain  $K_P$  must be specified. Refer to the data sheet of the corresponding device.

**Step 13.** In this step, the characteristic of the VCO will be determined (refer also to Fig. 5.2). As mentioned earlier, the procedure is restricted to relaxation VCOs. When a resonant VCO is used, however, the design can be adapted correspondingly (refer to the comments at the end of this chapter). Because the center frequency  $\omega_0$  (or the range of  $\omega_0$ ) is known and the range of the divider ratio  $N$  is also given, we can calculate the range of output (angular) frequencies that must be generated by the VCO. Let  $\omega_{2\min}$  and  $\omega_{2\max}$  be the minimum and maximum output frequencies of the VCO, respectively. A suitable VCO must be selected first. In most cases, the VCO will be part of the PLL system, typically an integrated circuit. Given the supply voltage(s) of the VCO, the data sheet indicates the usable range of control voltage  $u_f$ . For a unipolar power supply with  $U_B = 5$  V, this range is typically 1 to 4 V. Let  $u_{f\min}$  and  $u_{f\max}$  be the lower and upper limit of that range, respectively. The VCO is now designed such that it generates the output frequency  $\omega_2 = \omega_{2\min}$  for  $u_f = u_{f\min}$  and the output frequency  $\omega_2 = \omega_{2\max}$  for  $u_f = u_{f\max}$ . The corresponding VCO characteristic is plotted in Fig. 5.2. Now the VCO gain  $K_0$  is calculated from the slope of this curve—in other words

$$K_0 = \frac{\omega_{2\max} - \omega_{2\min}}{u_{f\max} - u_{f\min}}$$

Now the external components of the VCO can be determined also. Usually, the data sheets tell you how to calculate the values of these elements.

**Step 14.** Specify the type of loop filter. Because the PFD is used as a phase detector, the passive lead-lag filter will be used in most cases. This combination of phase detector and loop filter offers infinite pull-in range and hold range, as explained in Sec. 3.9.3. Other types of loop filters do not bring significant benefits, hence the following procedure is based on the passive lead-lag loop filter.

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

**Step 15.** Determination of dynamic properties of the PLL. To select an appropriate value for the natural frequency  $\omega_n$ , we must have an idea of how the PLL should react to dynamic events, for example, on frequency steps applied to the reference input, on a variation of the divider factor  $N$ , or the like. Specification of dynamic properties strongly depends on the intended use of the PLL system. Because different goals can be envisaged, this design step is a decision block having three outputs—in other words, you can specify dynamic performance of the PLL in three different ways.

In the first case, the PLL is used as a digital frequency synthesizer. Here, it could be desirable that the PLL switches very quickly from one output frequency  $f_{21}$  to another output frequency  $f_{22}$ . If the difference  $|f_{21} - f_{22}|$  is large, the PLL will probably lock out when switching from one frequency to the other. The user then would probably specify a maximum value for the pull-in time  $T_P$  the system needs to lock onto the new output frequency.  $T_P$  is then used to determine the remaining parameters of the PLL. If the user decides to specify  $T_P$  as a key parameter, the procedure continues at step 16.

In the second case, the PLL is also used as a digital frequency synthesizer. This synthesizer will generate integer multiples of a reference frequency  $f_{\text{ref}}$ —that is, the frequency at the output of the VCO is given by  $f_2 = N \cdot f_{\text{ref}}$ , where  $N$  is variable. In many synthesizer applications, it is desired that the PLL does not lock out if the output frequency changes from one frequency “channel” to an adjacent “channel,”—that is, if  $f_2$  changes from  $N_0 \cdot f_{\text{ref}}$  to  $(N_0 + 1) \cdot f_{\text{ref}}$ . In this case, the pull-out range  $\Delta\omega_{PO}$  is required to be less than  $f_{\text{ref}}$ . If the user decides to use  $\Delta\omega_{PO}$  as a key parameter, the procedure continues at step 20.

The third case of this decision step represents the more general situation where neither the pull-in time nor the pull-out range is of primary interest. Here the user must resort to a specification that makes as much sense as possible. Probably the simplest way is to make an assumption on the lock-in time  $T_L$  (also referred to as settling time) or even to specify the natural frequency immediately. If the third case is chosen, the design proceeds with step 21.

**Step 16.** When the voltage output PFD is used, the sum of both time constants  $\tau_1 + \tau_2$  of the loop filter is computed from the equation for  $T_P$  in [Table 3.4](#). For  $\Delta\omega_0$ , the maximum (radian) frequency step at the VCO output must be entered.

When the current output PFD is used, we will compute the value of capacitor  $C_1$  from the equation for  $T_P$  in [Table 3.5](#). For  $\Delta\omega_0$ , the maximum (radian) frequency step at the VCO output must be entered as well. The design proceeds with step 17.

**Step 17.** When the voltage output PFD is used, the natural frequency  $\omega_n$  is now computed from the sum  $\tau_1 + \tau_2$  using the equation for  $\omega_n$  in [Table 3.4](#). When the current output PFD is used, however, we will calculate the natural frequency  $\omega_n$  from the known parameters  $K_P$ ,  $K_0$ ,  $N$ , and  $C_1$ , using the equation for  $\omega_n$  in [Table 3.5](#). The design proceeds with step 18.

**Step 18.** First we describe the design procedure for the case where the voltage output PFD is used. Given  $\omega_n$  and  $\zeta$  now, time constant  $\tau_2$  is calculated using

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

the formula for  $\zeta$  given in [Tables 3.4](#). Because the sum  $\tau_1 \geq \tau_2$  has been computed in step 16,  $\tau_1$  can now be determined. Strange things may happen at this point: we perhaps obtained  $\tau_1 \geq \tau_2 = 300 \mu\text{s}$  in a previous step and computed  $\tau_2 = 400 \mu\text{s}$  right now! To realize the intended system, we would have to choose  $\tau_1 = -100 \text{ ms}$ , which is impossible. Whenever we meet that situation, the system cannot be realized with the desired goals (that is, with the desired values for  $\omega_n$  and/or  $\zeta$ ). It only becomes realizable if we choose a lower  $\omega_n$ , for example, which increases  $\tau_1 \geq \tau_2$ , or if we specify a lower  $\zeta$ , which decreases  $\omega_2$ . These odd things cannot happen in case of the current output PFD. Here we simply have to compute  $\tau_2$  using the equation for  $z$  in [Table 3.5](#).

**Step 19.** Given  $\tau_1$  and  $\tau_2$  (plus eventually  $K_a$ ), the components of the loop filter can be determined. We first discuss the procedure for the case where not a current-output PFD is applied. If the passive lead-lag filter is chosen, we have  $R_1 C_1 = \tau_1$  and  $R_2 C_1 = \tau_2$  (refer to [Fig. 2.17a](#)).  $C_1$  can be chosen arbitrarily; it should be specified such that we get “reasonable” values for  $R_1$  and  $R_2$ —in other words, in the range of kOhms to MOhms. For the active lead-lag filter ([Fig. 2.19a](#)), we have  $K_a = C_1/C_2$ ,  $\tau_1 = R_1 C_1$ , and  $\tau_2 = R_2 C_2$ . Here, we should first specify  $C_1$  in order to get a reasonable value for  $R_1$ . Then we would compute  $C_2$  from  $K_a = C_1/C_2$ , and finally we would compute  $R_2$  from  $\tau_2 = R_2 C_2$ . For the active PI filter ([Fig. 2.21a](#)), we have  $R_1 C_1 = \tau_1$  and  $R_2 C_1 = \tau_2$ . As with the passive lead-lag filter, we would first select  $C_1$  to get reasonable values for  $R_1$  and  $R_2$ .

When the current output PFD is used, capacitor value  $C_1$  has already been computed in step 17 or 22. Only resistor  $R_2$  has to be determined now from  $\tau_2 = C_1 R_2$ .

**Step 20.** The procedure is identical for both types of PFD. Given the pull-out range  $\Delta\omega_{PO}$  and damping factor  $\zeta$ , the formula for  $\Delta\omega_{PO}$  in [Table 3.4](#) or [3.5](#) is used to calculate the natural frequency  $\omega_n$ . The design proceeds with step 22.

**Step 21.** The procedure is identical for both types of PFD. Given the lock-in time  $T_L$ , the natural frequency  $\omega_n$  is calculated from the formula for  $T_L$  in Tables [3.4](#) or [3.5](#). The procedure continues with step 22.

**Step 22.** When the voltage output PFD is used, the sum of both time constants  $\tau_1 \geq \tau_2$  is computed from the given natural frequency  $\omega_n$ , using the equation for  $\omega_n$  in [Table 3.4](#). When the current output PFD is used, however, we are going to calculate the value of capacitor  $C_1$  from the given natural frequency  $\omega_n$ , using the equation for  $\omega_n$  in [Table 3.5](#). The design proceeds with step 18.

To make things more transparent, we will apply the design procedure presented here to the layout of an integer-N frequency synthesizer in [Chap. 6](#).

**Some comments on resonant tank VCOs, VCXOs, and the like.** As mentioned earlier, relaxation VCOs cannot operate at frequencies above about 50 MHz. In applications involving higher frequencies, the resonant type VCO must be used—therefore, refer to [Fig. 2.26](#). In the following, design equations are given that are required to specify the components of this type

of VCO. Assume for the

moment that the series capacitor  $C_S$  has a very high value; thus, the varactor capacitance appears in parallel to the capacitance of the series connection of  $C_1$  and  $C_2$ . Let  $C = \frac{C_1 C_2}{C_1 + C_2}$ , then the total capacitance of the tank circuit becomes  $C + C_V$ , where  $C_V$  is the capacitance of the varactor. The component values are now chosen such that the oscillator generates the output (radian) frequency  $\omega_{2\min}$  when the loop filter output voltage is  $u_{f\min}$ , and the output frequency is  $\omega_{2\max}$  when  $u_f = u_{f\max}$ . For  $u_f = u_{f\min}$ , the varactor capacitance is supposed to be  $C_{v\max}$ , while for  $u_f = u_{f\max}$ , the varactor capacitance is supposed to be  $C_{v\min}$ . Afterward, the maximum output frequency is determined by

$$\omega_{2\max} = \frac{1}{\sqrt{(C + C_{v\min})L}} \quad (5.6)$$

and the minimum output frequency is

$$\omega_{2\min} = \frac{1}{\sqrt{(C + C_{v\max})L}} \quad (5.7)$$

Afterward, the ratio of maximum to minimum output frequency is given by

$$\frac{\omega_{2\max}}{\omega_{2\min}} = \sqrt{\frac{C + C_{v\max}}{C + C_{v\min}}} \quad (5.8)$$

Generally, these ratios are close to 1—for example, 1.01. This means the varactor capacitance is small compared with capacitance  $C$ . Introducing the variables  $\Delta\omega_2 = \omega_{2\max} - \omega_{2\min}$  and  $\Delta C_V = C_{v\max} - C_{v\min}$ , we have approximately

$$1 + \frac{\Delta\omega_2}{\omega_{2\min}} \approx 1 + \frac{\Delta C_V}{C} \quad (5.9)$$

Now we introduce the relative quantities  $\Delta\omega_{2\text{rel}} = \frac{\Delta\omega_2}{\omega_{2\min}}$ ,  $\Delta C_{V\text{rel}} = \frac{\Delta C_V}{C}$ , and get the expression

$$1 + \Delta\omega_{2\text{rel}} \approx \sqrt{1 + \Delta C_{V\text{rel}}} \quad (5.10)$$

Because the relative quantities are much smaller than 1, we can apply a Taylor series expansion to Eq. (5.10) and finally get

$$\frac{\Delta C_V}{C} \approx 2 \frac{\Delta\omega_2}{\omega_{2\min}} \quad (5.11)$$

In a practical design, the variation of varactor capacitance  $\Delta C_V$  is given. [Equation \(5.11\)](#) then sets an upper limit to capacitance  $C$ . When  $C$  is chosen larger, the required output frequency range  $\Delta\omega_2$  cannot be realized. If [Eq. \(5.11\)](#)

---

results in a high value for  $C$  that is difficult to realize, we can choose a smaller value of  $C$ . However, because the output frequency range would become too large, we must select a smaller series capacitor  $C_S$  (Fig. 2.26) in order to reduce the capacitance variation by the varactor. A design procedure for the VCO could look like the following:

1. Determine minimum output frequency  $\omega_{2\min}$ , output frequency range  $\Delta\omega_2$ , and varactor capacitance variation  $\Delta C_V$ .
2. Using Eq. (5.11), set a suitable value for  $C$ .
3. Determine inductor  $L$  such that the oscillator frequency is  $\omega_{2\min}$  for  $C_V = C_{V\max}$ .
4. When  $C$  has been chosen smaller than the value resulting from Eq. (5.11), set  $C_S$  such that the capacitance variation is large enough to produce the desired output frequency range.

In many applications, the tank circuit of the oscillator is a quartz crystal. The resonant frequency of a quartz oscillator can be pulled over a restricted range by a varactor diode in parallel. To design such an oscillator, the crystals specifications must be known; the data sheet will tell you how much the resonant frequency is pulled away when switching a capacitor in parallel to the crystal.

At frequencies in the GHz region, discrete inductors become difficult to implement. Moreover, they offer very low  $Q$  factors, approximately in the range of 2 to 3. To get better resonators, the discrete LC tank circuit is frequently replaced by a transmission line stub, for example, a piece of microstrip. Radial stub lines can also be used.<sup>53</sup> Very detailed information for the design of high-frequency oscillators can be found, for example, in the publications of Rohde.<sup>11, 48, 49</sup>

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

# Mixed-Signal PLL Applications Part 1: Integer-N Frequency Synthesizers

## Synthesizers in Wireless and RF Applications

PLL frequency synthesizers play an ever-increasing role in the field of communications. Originally, the frequency synthesizer has been a system creating a set of frequencies that were an integer multiple of a mostly fixed reference frequency. Such synthesizers (referred to as *integer-N frequency synthesizers*) are found in every FM radio receiver, every TV receiver, and the like. Later the fractional-*N* synthesizer was developed. In contrast to the integer-*N* frequency synthesizer, this novel device is able to create frequencies that are  $N.f$  times a reference frequency, where  $N$  is the integer part and  $f$  is the fractional part an arbitrary number  $N.f$ . Whereas fractional-*N* synthesizers have been considered rather “exotic” in the past, they suddenly have gained increased interest, mainly in mobile communications and in spread-spectrum applications. Fractional-*N* frequency synthesizers will be discussed in greater detail in [Chap. 7](#).

Conventional communications used one single carrier whose frequency was fixed. Radio and TV transmitters are examples for this category. In military communications, it showed up that such constant carrier frequency links could easily be corrupted by “jammers.” <sup>20</sup> This lead to the development of “frequency hopping.” In frequency-hopping applications, the single carrier is replaced by a large set of carrier frequencies. This set consists of a number  $N$  of carrier frequencies that are switched in a pseudo-random manner. This means the transmitter repeatedly jumps through these  $N$  carrier frequencies. The receiver, which must know the carrier frequency sequence of course, has to track the carrier frequency at any time. Each individual carrier frequency is called a “chip” in the frequency-hopping vocabulary. The duration of such a “chip” is usually on the order of several hundred microseconds. This implies that the receiver must be able to switch the carrier frequency quickly—say, within about  $100\ \mu s$ . It will be demonstrated in [Chap. 7](#) that fractional-*N* synthesizers

can switch their output frequency more rapidly than conventional integer- $N$  synthesizers.

## Integer- $N$ Frequency Synthesizers without Prescalers

A very simple frequency synthesizer has already been shown in [Fig. 2.1](#). This circuit is redrawn in [Fig. 6.1](#), where it is shown that the scaling factor  $N$  is usually set by an external digital signal. In [Fig. 6.1](#), this control signal has been plotted as a parallel digital input; in many frequency synthesizers, this signal can also be serial. In this drawing, the reference frequency is denoted as  $f_1$ . In this simple arrangement, the VCO creates an output frequency  $f_2$ , which is simply  $N \cdot f_1$ .

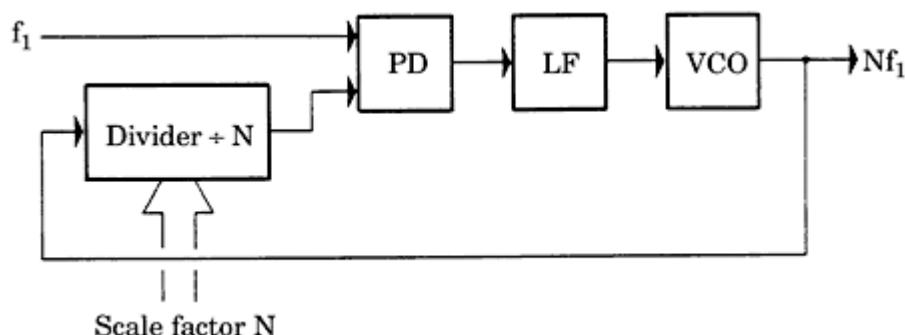
Frequency synthesizers are found in FM receivers, CB transceivers, television receivers, and the like. In these applications, there is a need for generating a great number of frequencies with a narrow spacing of 50, 25, 10, 5, or even 1 kHz. If a channel spacing of 10 kHz is desired, a reference frequency of 10 kHz is normally chosen. Most oscillators are quartz-crystal stabilized. A quartz crystal oscillating in the kilohertz region is quite a bulky component. It is therefore more convenient to generate a higher frequency, typically in the region of 5 to 10 MHz, and to scale it down to the desired reference frequency. In most of the frequency-synthesizer ICs presently available, a reference divider is integrated into the chip, as shown in [Fig. 6.2](#).

The oscillator circuitry is also included on most of these ICs. When the scaling factor of the reference divider is denoted  $R$  and the scaling factor of the other divider  $N$ , the VCO creates an output frequency given by

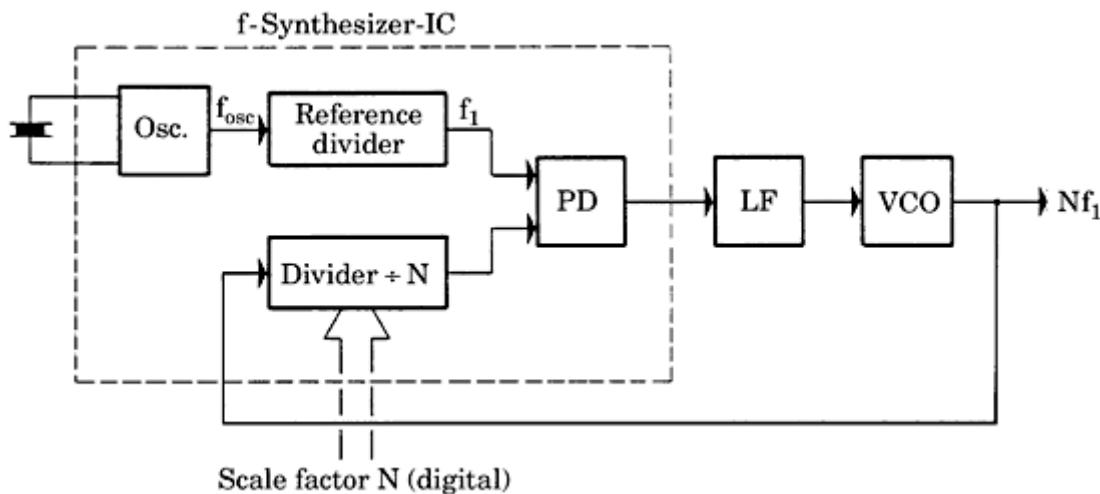
$$f_2 = \frac{N}{R} f_{\text{osc}} = N \cdot f_1 \quad (6.1)$$

where  $f_{\text{osc}}$  is the frequency of the oscillator.

One seeks to include as many functions on the chip as possible. It is no major problem to implement all the digital functions on the chip, such as oscillators,



**Figure 6.1** A basic frequency synthesizer system.



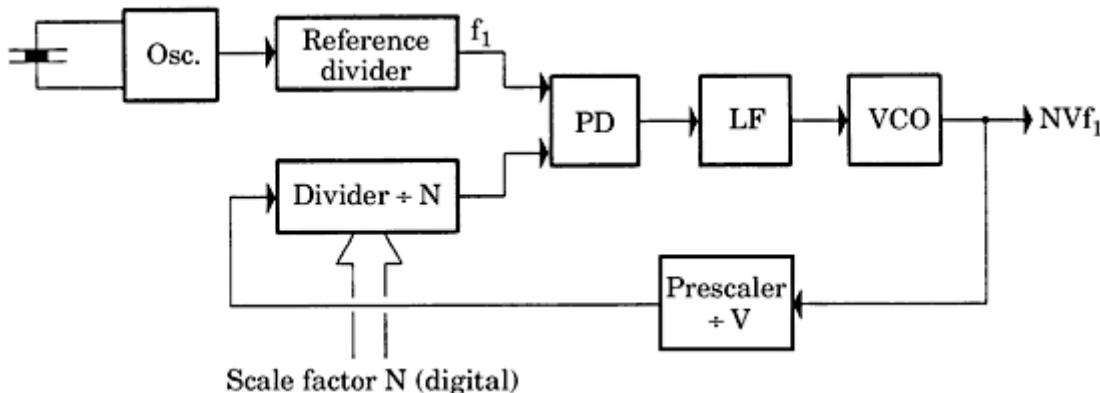
**Figure 6.2** A system equal in performance to Fig. 6.1 but with an additional reference divider that makes it possible to use a higher-frequency reference, normally a quartz oscillator.

phase detectors, frequency dividers, and so on, as indicated by the dashed enclosure in Fig. 6.2. Due to its low power consumption, high noise immunity, and large range of supply voltages, CMOS is the preferred technology today. The limited speed of CMOS devices precludes their application for *directly* generating frequencies in the range of 100 MHz or more (at least at the time of this writing).

## Integer-N Frequency Synthesizers with Prescalers

### Fixed division ratio prescalers

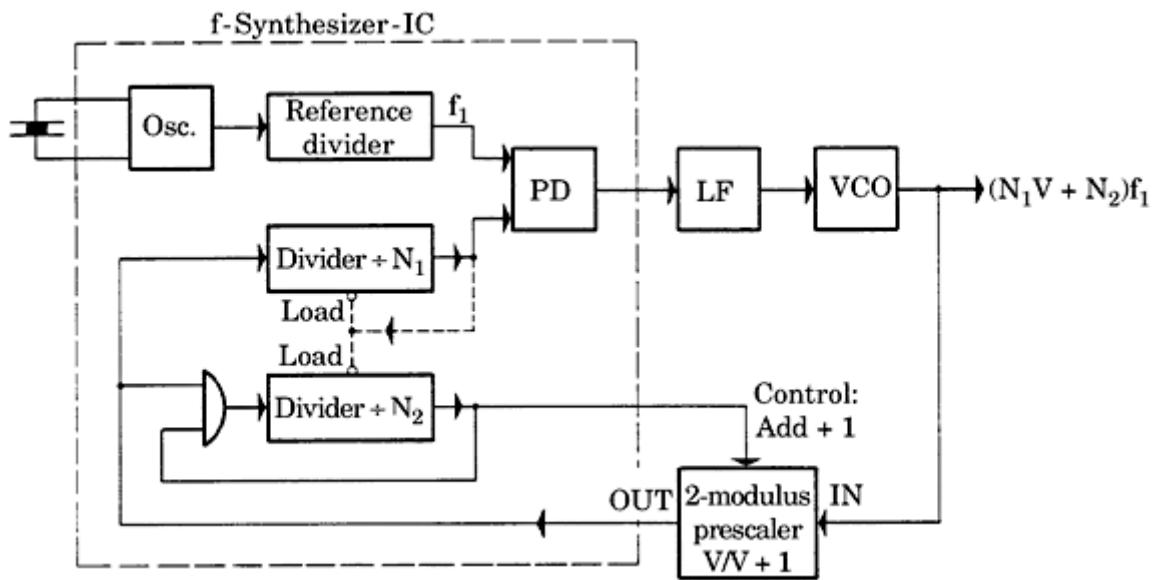
To generate higher frequencies, *prescalers* are used; these are often built with other IC technologies such as ECL, Schottky TTL, GaAs (gallium-arsenide), or SiGe (silicon-germanium compound) (see Fig. 6.3). Such prescalers extend the range of frequencies into the microwave frequency bands.<sup>7,37</sup>



**Figure 6.3** A frequency synthesizer extending the upper frequency range by using an additional high-speed prescaler. The channel spacing is increased to  $V \cdot f_1$ .

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.



**Figure 6.4** A frequency synthesizer using a dual-modulus prescaler. The channel spacing becomes  $f_1$ .

If the scaling factor of the prescaler is  $V$  (Fig. 6.3), the output frequency of the synthesizer becomes

$$f_{\text{out}} = NVf_1$$

Obviously, the scaling factor  $V$  of the prescaler is much greater than 1 in most cases. This implies that it is no longer possible to generate every desired integer multiple of the reference frequency  $f_1$ ; if  $V$  is say, 10, only output frequencies of  $10 \cdot f_1$ ,  $20 \cdot f_1$ ,  $30 \cdot f_1$ , and so on can be generated. This disadvantage can be circumvented by using a so-called dual-modulus prescaler, as shown in Fig. 6.4.<sup>11,38</sup>

### Dual-modulus prescalers

A dual-modulus prescaler is a counter whose division ratio can be switched from one value to another by an external control signal. As an example, the prescaler in Fig. 6.4 can divide by a factor of 11 when the applied control signal is HIGH, or by a factor of 10 when the control signal is LOW. It can be demonstrated that the dual-modulus prescaler makes it possible to generate a number of output frequencies that are spaced only by  $f_1$  and not by a multiple of  $f_1$ .

The following conventions are used with respect to Fig. 6.4:

- Both programmable  $\div N_1$  and  $\div N_2$  counters are DOWN counters.
- The output signal of both of these counters is HIGH if the content of the corresponding counters has not yet reached the value 0.
- When the  $\div N_1$  counter has counted down to 0, its output goes LOW and it immediately loads both counters to their preset values  $N_1$  and  $N_2$ , respectively.

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

- $N_1$  is always greater than or equal to  $N_2$ .
- As shown by the AND gate in [Fig. 6.4](#), underflow below 0 is inhibited in the case of the  $\div N_2$  counter. If this counter has counted down to 0, further counting pulses are inhibited.

The operation of the system shown in [Fig. 6.4](#) becomes clearer if we assume that the  $\div N_1$  counter has just counted down to 0 and both counters have been loaded with their preset values  $N_1$  and  $N_2$ , respectively. We now have to find the number of cycles the VCO must produce until the same logic state is reached again. This number is the overall scaling factor  $N_{\text{tot}}$  of the arrangement shown in [Fig. 6.4](#). As long as the  $\div N_2$  counter has not yet counted down to 0, the prescaler is dividing by  $V + 1$ . Consequently, both the  $\div N_1$  and the  $\div N_2$  counters will step down by one count when the VCO has generated  $V + 1$  pulses. The  $\div N_2$  counter will therefore step down to 0 when the VCO has generated  $N_2 \cdot (V + 1)$  pulses. At that moment, the  $\div N_1$  counter has stepped down by  $N_2$  counts—that is, its content is  $N_1 - N_2$ .

The scaling factor of the dual-modulus prescaler is now switched to the value  $V$ . The VCO will have to generate additional  $(N_1 - N_2)V$  pulses until the  $\div N_1$  counter steps to 0. When the content of  $N_1$  becomes 0, both the  $\div N_1$  and the  $\div N_2$  counters are reloaded to their preset values, and the cycle is repeated.

How many pulses  $N_{\text{tot}}$  did the VCO produce to run through one full cycle?  $N_{\text{tot}}$  is given by

$$N_{\text{tot}} = N_2(V + 1) + (N_1 - N_2)V$$

Factoring out yields the simple expression

$$N_{\text{tot}} = N_1V + N_2 \quad (6.2)$$

As stated earlier,  $N_1$  must always be greater than or equal to  $N_2$ . If this were not the case, the  $\div N_1$  counter would be stepped down to 0 *earlier* than the  $\div N_2$  counter, and both counters would then be reloaded to their preset values. The dual-modulus prescaler *never* would be switched from  $V + 1$  to  $V$ , so the system could not work in the intended way.

If  $V = 10$ , [Eq. \(6.2\)](#) becomes

$$N_{\text{tot}} = 10N_1 + N_2 \quad (6.3)$$

In this expression,  $N_2$  represents the units and  $N_1$  the tens of the overall division ratio  $N_{\text{tot}}$ . Then  $N_2$  must be in the range of 0 – 9, and  $N_1$  can assume any value greater than or equal to 9—that is,  $N_{1\text{min}} = 9$ . The smallest realizable division ratio is therefore

$$N_{\text{tot,min}} = N_{1\text{min}}V = 90$$

The synthesizer of [Fig. 6.4](#) is thus able to generate all integer multiples of the reference

frequency  $f_1$  starting from  $N_{\text{tot}} = 90$ .

Other factors can of course be chosen for  $V$ . If  $V = 16$ , the dual-modulus prescaler would divide by 16 or 17. The overall division ratio would then be

$$N_{\text{tot}} = 16N_1 + N_2$$

Now  $N_2$  would be required to have a range of 0 to 15, and the minimum value of  $N_1$  would be  $N_{1\text{min}} = 15$ . In this case, the smallest realizable division ratio  $N_{\text{tot,min}}$  would be 240.

Let us again assume that  $V$  is chosen at 10, and that the (scaled-down) reference frequency  $f_1$  of the system in Fig. 6.4 is 10 kHz. The smallest output frequency would then be  $90 \cdot f_1 = 900$  kHz.

For the circuits inside the dashed enclosure, CMOS devices are normally used. The counting frequency of older CMOS ICs (such as the old series 74Cxxx) has been limited to approximately 3 MHz, so when using these devices a maximum frequency of only about 30 MHz could be realized for a prescaler ratio of  $V = 10$ . To extend the frequency range, larger prescaler ratios, say  $V = 100$ , became desirable. Using  $V = 100$ , ratio  $N_{\text{tot}}$  would be

$$N_{\text{tot}} = 100N_1 + N_2$$

where  $N_2$  must now cover the range of 0 to 99, and  $N_1$  must be at least 99. It should be noted, however, that now the lowest division ratio  $N_{\text{tot,min}}$  is no longer 90 but has been increased to

$$N_{\text{tot}} = 100N_{1\text{min}} = 100 \cdot 99 = 9900$$

If the reference frequency  $f_1$  is still 10 kHz, the lowest frequency to be synthesized is now 99 MHz.

## Four-modulus prescalers

Fortunately, there is another way, which extends the upper frequency range of a frequency synthesizer but still allows the synthesis of lower frequencies. The solution is the *four-modulus prescaler* (Fig. 6.5). The four-modulus prescaler is a logical extension of the dual-modulus prescaler. It offers four different scaling factors, and two control signals are required to select one of the four available scaling factors.

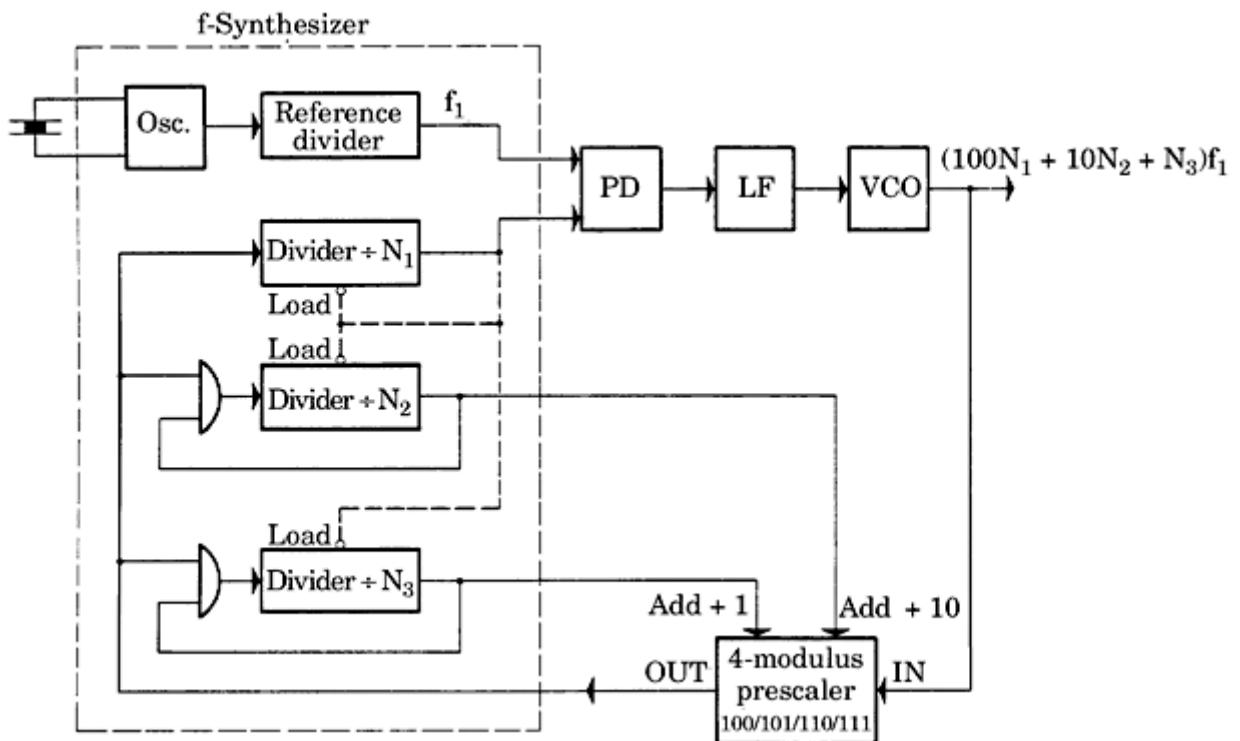
As an example, the four-modulus prescaler shown in Fig. 6.5 can divide by factors of 100, 101, 110, and 111.<sup>11,36</sup> By definition, it scales down by 100 when both control inputs are LOW. The internal logic of the four-modulus prescaler is designed so that the scaling factor is increased by 1 when one of the control signals is HIGH, or increased by 10 when the other control signal is HIGH. If both control signals are HIGH, the scaling factor is increased by  $1 + 10 = 11$ .

As seen in Fig. 6.5, there are no longer two programmable  $\div N$  counters in the system, but *three*:  $\div N_1$ ,  $\div N_2$ , and  $\div N_3$  dividers. The overall division ratio  $N_{\text{tot}}$  of this arrangement is given by

$$N_{\text{tot}} = 100N_1 + 10N_2 + N_3$$

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.



**Figure 6.5** A frequency synthesizer using a four-modulus prescaler. This extends the high-end of the frequency range, while allowing to create lower frequencies than obtainable from a synthesizer with a dual-modulus prescaler.

In this equation  $N_3$  represents the units,  $N_2$  the tens, and  $N_1$  the hundreds of the division ratio  $N_{\text{tot}}$ . Here  $N_2$  and  $N_3$  must be in the range 0 to 9, and  $N_1$  must be at least as large as both  $N_2$  and  $N_3$  for the reasons explained in the previous example ( $N_{1\min} = 9$ ).

The smallest realizable division ratio is consequently

$$N_{\text{tot},\min} = 100 \cdot 9 = 900$$

which is lower roughly by a factor of 10 than the previous example. For a reference frequency  $f_1$  of 10 kHz, the lowest frequency to be synthesized is therefore  $900 \cdot f_1 = 9$  MHz.

Let us examine the operation of the system in Fig. 3.5 by giving a numerical example.

**Numerical Example** We wish to generate a frequency that is 1023 times the reference frequency. The division ratio  $N_{\text{tot}}$  is thus 1023; hence  $N_1 = 10$ ,  $N_2 = 2$ , and  $N_3 = 3$  are chosen. Furthermore, we assume that the  $\div N_1$  counter has just stepped down to 0, so all three counters are now loaded to their preset values. Both outputs of the  $\div N_2$  and  $\div N_3$  counters are now HIGH, a condition that causes the four-modulus prescaler to divide initially by 111.

**Solution** After  $N_2 \cdot 111 = 2 \cdot 111 = 222$  pulses generated by the VCO, the  $\div N_2$  counter steps down to 0. Consequently, the prescaler will divide by 101. At this moment, the content of the  $\div N_3$  counter is  $3 - 2 = 1$ . After another 101 pulses have been generated by the VCO,

the  $\div N_3$  counter also steps down to 0. The division ratio of the four-modulus prescaler is now 100.

The content of the  $\div N_1$  counter is now 7. After another 700 pulses have been generated by the VCO, the  $\div N_1$  counter also steps down to 0, and the cycle is repeated. To step through an entire cycle, the VCO had to produce a total of

$$N_{\text{tot}} = 2 \cdot 111 + 1 \cdot 101 + 7 \cdot 100 = 1023$$

pulses, which is exactly the number desired.

## Extending the Frequency Range with Mixers and Frequency Multipliers

In all frequency synthesizer systems previously considered, multiples of a reference frequency have been generated exclusively by scaling down the VCO output signal by various counter configurations. To produce frequencies in the range of 98.7 to 118.7 MHz with a spacing of 100 kHz, a synthesizer circuit would have had to be designed to offer an overall division ratio of 987 to 1187. As an alternative, one could first generate output frequencies in the range of 8.7 to 18.7 MHz, using a division ratio of 87 to 187, and then *mix up* the obtained frequency band to the desired band. An additional local oscillator operating at a frequency of 90 MHz would be required in this case.

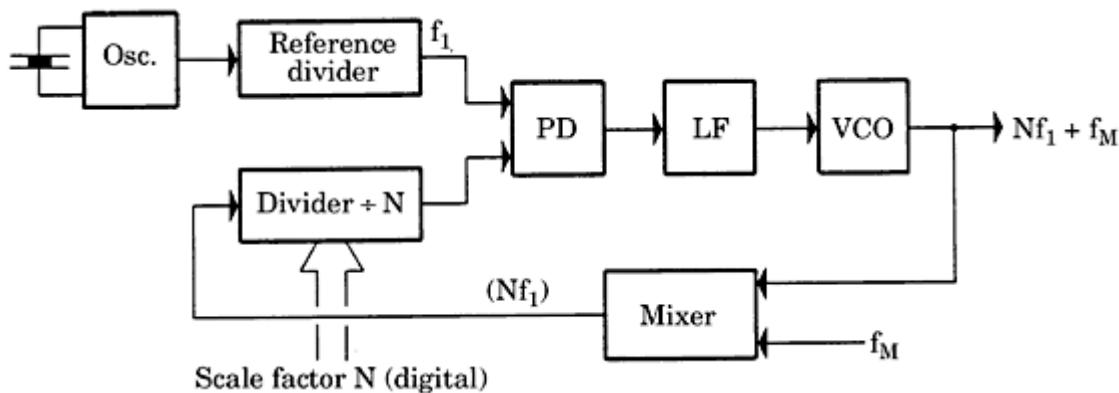
A frequency synthesizer system using an up-mixer is shown in Fig. 6.6. The basic synthesizer circuit employed here corresponds to the simple system shown in Fig. 6.2. Of course, all synthesizer systems using dual- and four-modulus prescalers can be combined with a mixer. In the system in Fig. 6.6, the frequency of the local oscillator is  $f_M$ . Consequently, the synthesizer produces output frequencies given by

$$f_{\text{out}} = Nf_1 + f_M$$

The mixer is used here to *mix down* these frequencies to the *baseband*  $N \cdot f_1$ . The mixer also generates a number of further mixing products effectively, generally frequencies given by

$$f_{\text{mix}} = \pm nf_{\text{out}} \pm mf_M$$

where  $n$  and  $m$  are arbitrary positive integers.

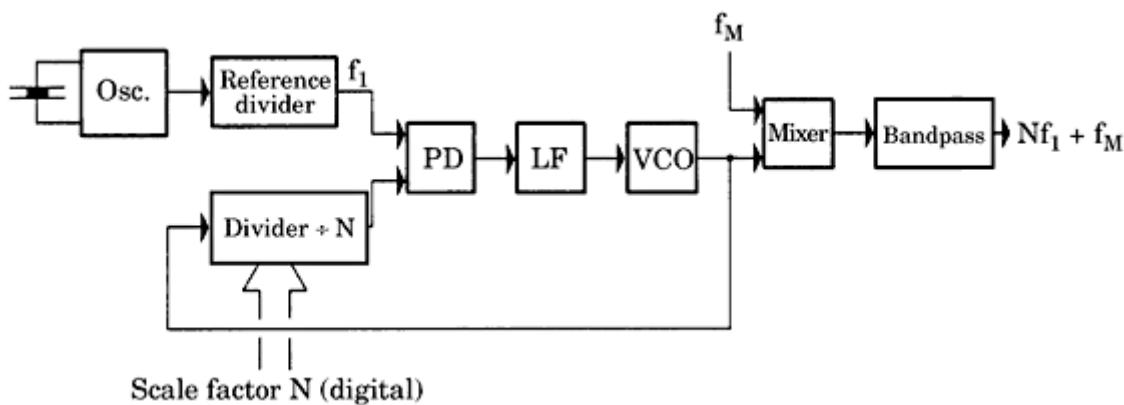


**Figure 6.6** A frequency synthesizer with a mixer to extend the high-end of the frequency

range.

---

**Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.**



**Figure 6.7** Frequency synthesizer using a mixer to extend the upper end of the frequency range. In contrast to the circuit shown in Fig. 6.6, here the mixer is outside the loop.

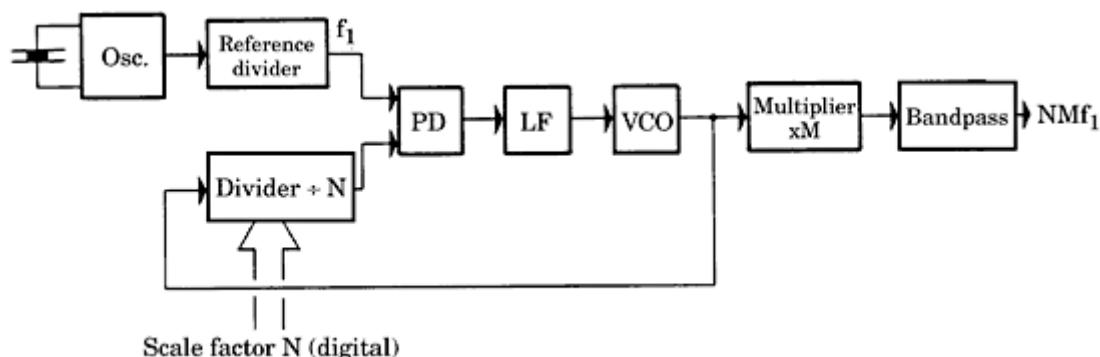
All these mixing products (excluding the baseband  $f_{\text{out}} - f_M$ ) have frequencies that are very much higher than the baseband, so they are filtered easily by either a low-pass filter or even the PLL system itself.

An alternative arrangement using a mixer is shown in Fig. 6.7. In contrast to the previously discussed system, the mixer is not inside but outside the loop. The system generates output frequencies identical with those in Fig. 6.6, but for obvious reasons the desired frequency spectrum has to be filtered out here by a bandpass filter.

Still another way of extending the upper frequency range of frequency synthesizers is given by the *frequency multiplier*, as shown by Fig. 6.8. Frequency multipliers are normally built from nonlinear elements that produce harmonics, such as varactor diodes, step-recovery diodes, and similar devices. These elements produce a broad spectrum of harmonics. The desired frequency must therefore be filtered out by a bandpass filter. If  $M$  is the frequency-multiplying factor, the output frequency of this synthesizer is

$$f_{\text{out}} = MNf_1$$

It should be noted that now the channel spacing is not equal to the reference frequency  $f_1$ , but to  $M \cdot f_1$ .



**Figure 6.8** A frequency synthesizer using a frequency multiplier to extend the upper end of

the frequency range.

To illustrate the theory of frequency synthesizers we now will design an actual system that uses the design procedure described in [Chap. 5](#) and shown in the flowchart in [Fig. 5.1](#).

## Case Study: Designing an Integer-N PLL Frequency Synthesizer

The frequency synthesizer is required to produce a set of frequencies in the range from 1 to 2 MHz with a channel spacing of 10 kHz—meaning frequencies of 1000, 1010, 1020, and so on up to 2000 kHz will be generated. For this design, we use the popular 74HC/HCT4076 CMOS device which is based on the old industry standard CD 4046 originally introduced by RCA. This circuit contains three different phase detectors, an EXOR gate, a JK-flipflop, and a PFD (with voltage output). Because noise must not be considered in this design, we use the PFD as phase detector. Because this detector offers infinite pull-in range for any type of loop filter, we use the simplest of these, the passive lead-lag. The supply voltage  $U_B$  is chosen as 5 V. With these assumptions, we are ready to start the design, following the procedure shown in [Fig. 5.1](#) ([Chap. 5](#)).

**Step 1.** Determine ranges of input and output frequencies. The input frequency is constant,  $f_1 = 10$  kHz. The output frequency is in the range of 1 to 2 MHz; thus,  $f_{2\min} = 1$  MHz,  $f_{2\max} = 2$  MHz.

**Step 2.** The divider ratio must be variable in the range  $N = 100$  to 200. The PLL will be optimized ( $\zeta = 0.7$ ) for the divider ratio  $N_{\text{mean}} = \sqrt{N_{\min}N_{\max}} = 141$ , as will be shown in step 3.

**Step 3.** Determination of damping factor  $\zeta$ . Selecting  $\zeta = 0.7$  at  $N = N_{\text{mean}}$  yields the following minimum and maximum values for  $\zeta$ :

$$\zeta_{\min} = 0.59 \quad \text{for } N = 200$$

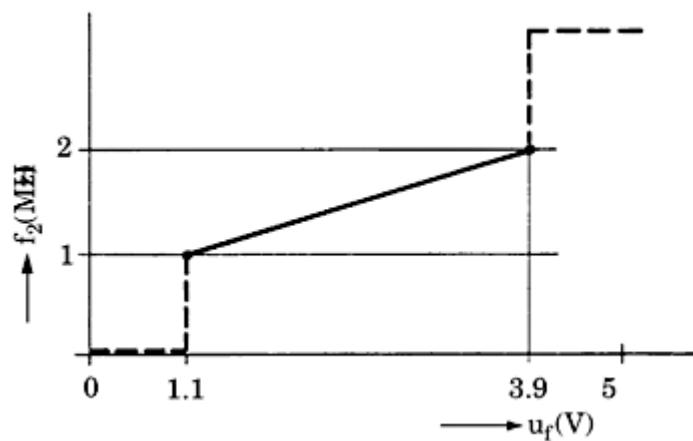
$$\zeta_{\max} = 0.83 \quad \text{for } N = 100$$

This range is acceptable.

**Step 4.** Noise is not of concern in this PLL design, so the procedure continues with step 12.

**Step 12.** Selection of the phase detector type. The PFD (with voltage output) is chosen, as noted in the introductory remarks. The phase detector gain becomes  $K_d = 5/4\pi = 0.4$  V/rad.

**Step 13.** VCO layout. According to the data sheet of the 74HC4046A IC, the VCO operates linearly in the voltage range of  $u_f = 1.1$  to 3.9 V approximately. Therefore, the characteristic of [Fig. 6.9](#) can be plotted. If the VCO input voltage exceeds about 3.9 V, the VCO generates a very high frequency (around 30 MHz); if it falls below 1.1 V, the VCO frequency is extremely low—in other words, only



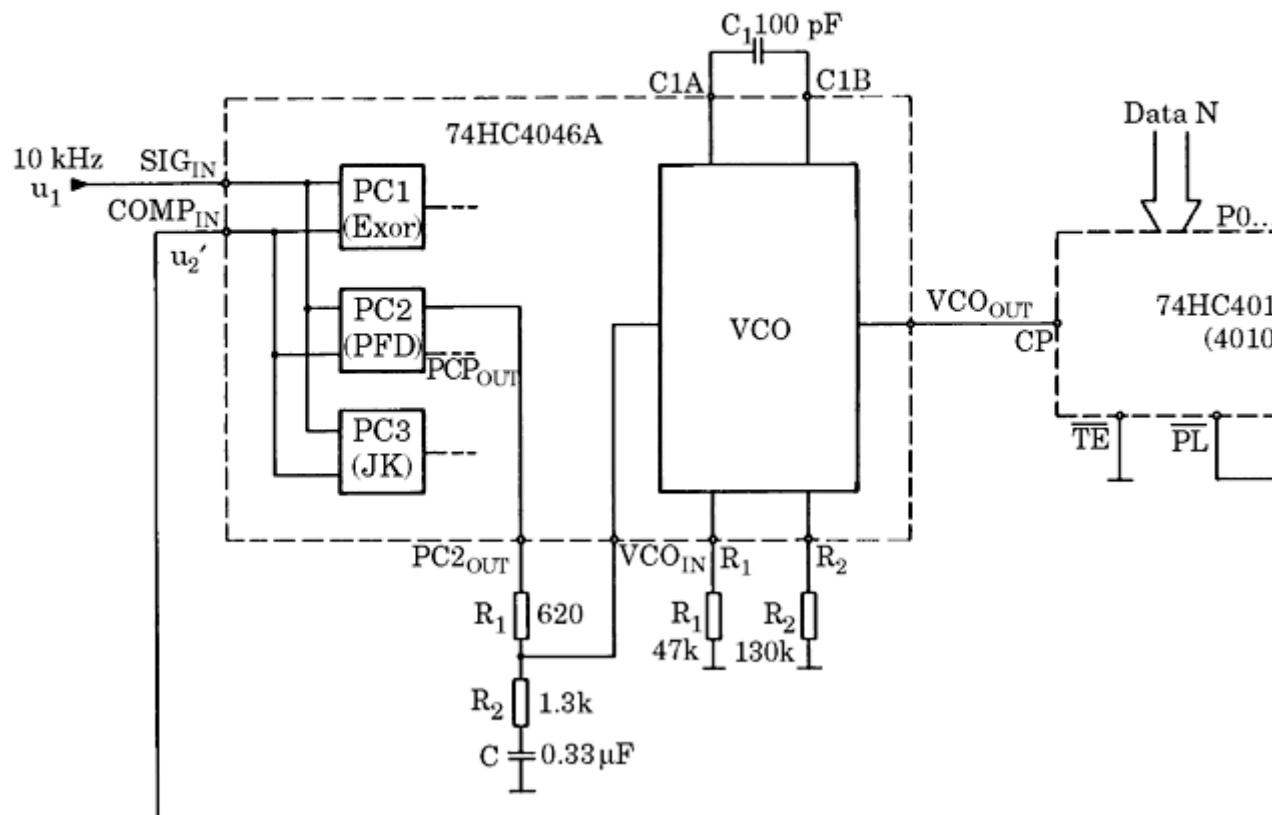
**Figure 6.9** Characteristic of the VCO for the CMOS IC type 74HC/HCT4046.

a few Hertz. From this characteristic, the VCO gain becomes  $K_0 = 2.24 \cdot 10^6 \text{ rad s}^{-1}\text{V}^{-1}$ . Following the rules indicated in the data sheet, resistors  $R_1$ ,  $R_2$ , and capacitor  $C_1$  ([Fig. 6.10](#)) are found from graphs. The resistors must be chosen to be in the range of 3 to 300 k $\Omega$ . The parallel connection of  $R_1$  and  $R_2$  should furthermore yield an equivalent resistance of more than 2.7 k $\Omega$ . We thus obtain

$$R_1 = 47 \text{ k}\Omega$$

$$R_2 = 130 \text{ k}\Omega$$

$$C_1 = 100 \text{ pF}$$



**Figure 6.10** Schematic diagram of the PLL frequency synthesizer.

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
 Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
 Any use is subject to the Terms of Use as given at the website.

**Step 14.** Loop filter selection. As indicated earlier, we choose the passive lead-lag filter.

**Step 15.** Here we must make some assumptions on dynamic behavior of the PLL. It is reasonable to postulate that the PLL should lock within a sufficiently short time—for example, within 2 ms. Hence, we set  $T_L = 2$  ms. The procedure continues with step 21.

**Step 21.** Given  $T_L$ , the natural frequency  $\omega_n$  is calculated:

$$\omega_n = \frac{2\pi}{T_L} = 3140 \text{ s}^{-1}$$

The procedure continues with step 22.

**Step 22.** Because a passive loop filter is used, the formula for  $\omega_n$  in [Table 3.4](#) can only be used to calculate the sum  $\tau_1 + \tau_2$ . We obtain

$$\tau_1 + \tau_2 = 644 \mu\text{s}$$

The procedure continues with step 18.

**Step 18.** Given  $\omega_n$ , we use the formula for  $\zeta$  in [Table 3.4](#) to calculate  $\tau_2$ . We obtain

$$\tau_2 = 445 \mu\text{s}$$

Now the time constant  $\tau_1$  can be computed. Because  $\tau_1 + \tau_2 = 644 \mu\text{s}$  (from step 22) and  $\tau_2 = 455 \mu\text{s}$ ,  $\tau_1$  becomes

$$\tau_1 = 199 \mu\text{s}$$

**Step 19.** Calculation of loop filter components. Given  $\tau_1$  and  $\tau_2$ , the loop filter components  $R_1$ ,  $R_2$ , and  $C$  can be determined. For optimum sideband suppression, capacitor  $C$  should be chosen as large, and resistors  $R_1$  and  $R_2$  as low, as possible. Selecting  $C = 0.33 \mu\text{F}$  gives the resistors (rounded to the next values of the R24 series)

$$R_1 = 620 \Omega$$

$$R_2 = 1.3 \text{ k}\Omega$$

The sum of  $R_1$  and  $R_2$  is higher than the minimum allowable load resistance ( $470 \Omega$ ).

The final design is shown in [Fig. 6.10](#). The 74HC4046A PLL contains three phase detectors: PC1 (EXOR), PC2 (PFD), and PC3 (JK-flipflop). PC2 has two outputs:  $\text{PC2}_{\text{OUT}}$  and  $\text{PCP}_{\text{OUT}}$ .  $\text{PC2}_{\text{OUT}}$  is the phase PFD output, and  $\text{PC2}_{\text{OUT}}$  is an in-lock detection signal—that is, a logical signal which becomes “high” when the PLL has acquired lock.

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

Only the PC2<sub>OUT</sub> is used in this application. For the down scaler, an eight-bit presetable down counter of type 74HC40102 or 74HC40103 is used. The 74HC40102 consists of two cascaded BDC counters, whereas the 74HC40103 is a binary counter. When the PE (preset enable) input is pulled “low,” the data on input port P0 through P7 are loaded into the counter. The counter counts down on every positive transition at the CP input (count pulse). If the counter has counted down to 0, the TC output (terminal count) goes low. Connecting TC with PE forces the counter to reload the data on the next counting pulse. If  $N$  is the number represented by the data bus, the counter divides by  $N + 1$  (and not by  $N$ ). To scale down by a factor of 100, for example, we must therefore apply  $N = 99$  to the input port.

As mentioned earlier, the Philips company provides a diskette with a design program for this type of PLL IC.<sup>52</sup> (This program can also be downloaded from [www.nxp.com](http://www.nxp.com), a website maintained by the Philips company.) The author repeated the design using this program and got design parameters very similar to those obtained by his own procedure.

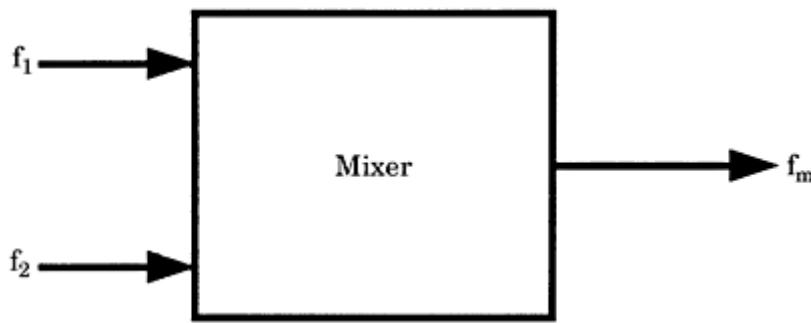
## Single-Loop and Multi-Loop Frequency Synthesizers

In Secs. 6.2 and 6.3, we exclusively considered frequency synthesizers which were built from one single loop. Such synthesizers were capable of creating a set of frequencies that were an integer multiple of a given reference frequency. We will see in this section that single-loop synthesizers can become impractical when it comes to generating a large range of frequencies with a very small channel spacing.

Think, for example, of a synthesizer which would be required to generate frequencies in the range of 100 to 200 MHz with a channel spacing of 1 kHz. Of course, we could implement a synthesizer as shown in Fig. 3.1, having a reference frequency of 1 kHz and using a divide-by- $N$  counter having a scaling factor  $N$  in the range of 100,000 to 200,000. Such a system would show up two flaws: first of all, it would be slow because it needs about 10 to 20 reference cycles to switch from one channel to another—that is, it would settle in perhaps 20 ms. Second, it would probably have poor noise performance. As will be shown in Sec. 6.7, the phase jitter at the output of the VCO increases with the square of the scaling factor  $N$ .

Another idea would be to use two separate synthesizers, a “course” synthesizer creating the frequency range from 100 to 200 MHz in steps of 1 MHz, and a “fine” synthesizer generating frequencies in the range of 0 to 1 MHz in steps of 1 kHz. (We will see later that the “fine” synthesizer must not necessarily be slow.) To add coarse and fine frequencies, we would use a mixer (cf. Fig. 6.11).

An extremely simple numerical example will demonstrate, however, that this simple arrangement would not work as expected. Assume, for example, that the coarse frequency  $f_1$  is 100 MHz and the fine frequency  $f_2$  is 1 kHz. An ideal mixer would multiply both input signals; hence, at its output we would have an upper



**Figure 6.11** Using a mixer to add a “coarse” frequency  $f_1$  and a “fine” frequency  $f_2$ .

sideband with frequency  $f_1 + f_2 = 100.001$  MHz and a lower sideband with frequency  $f_1 - f_2 = 99.999$  MHz. Because we only want to keep the upper sideband, we would have to filter out the lower. Such a filter is practically impossible to realize; it should pass the upper frequency, but reject the lower, hence the transition region of the filter would have to be extremely narrow! There is still another reason why the circuit in Fig. 6.11 would not work. We decided that both frequencies  $f_1$  and  $f_2$  should be variable. If we switch, for example,  $f_1$  to 130 MHz and  $f_2$  to 5 kHz, the filter now had to separate the sidebands 130.005 MHz and 129.995 MHz.

In order to separate upper and lower sidebands, we will have to introduce a frequency offset at the  $f_2$  input. Instead of applying  $f_2$  directly, we would apply another “fine” frequency  $f_2'$ , which is given by  $f_2' = f_2 + f_{\text{ofs}}$ , where  $f_{\text{ofs}}$  is the frequency offset. How large must  $f_{\text{ofs}}$  be chosen to safely separate upper and lower sidebands at the mixer output?

Let the minimum coarse frequency be  $f_{1\min}$  and the maximum coarse frequency  $f_{1\max}$ . In analogy, let the minimum fine frequency be  $f_{2\min}$  and the maximum  $f_{2\max}$ . Then, the maximum frequency in the upper sideband would be

$$f_{u\max} = f_{1\max} + f_{\text{ofs}} + f_{2\max}$$

and the minimum frequency in the upper sideband

$$f_{u\min} = f_{1\min} + f_{\text{ofs}} + f_{2\min}$$

For the lower sideband, the maximum and minimum frequencies would be

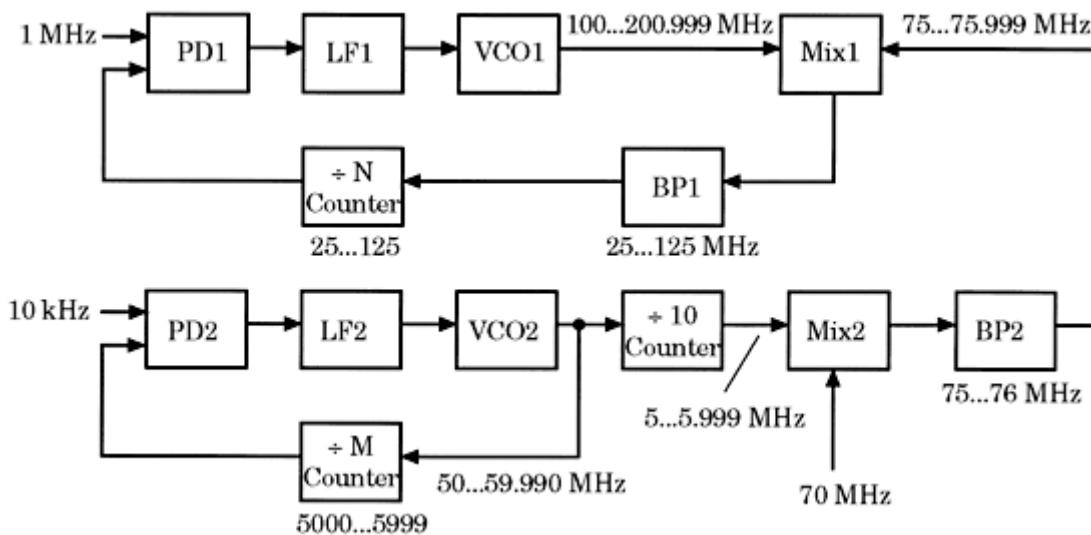
$$f_{l\max} = f_{1\max} - f_{\text{ofs}} - f_{2\min}$$

$$f_{l\min} = f_{1\min} - f_{\text{ofs}} - f_{2\max}$$

To separate the sidebands, the minimum frequency of the upper sideband must be larger than the maximum frequency of the lower sideband; hence, we have the condition

$$f_{u\min} > f_{l\max}$$

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.



**Figure 6.12** A dual-loop frequency synthesizer built from a synthesizer creating coarse frequency steps (1 MHz, upper part) and from another synthesizer creating fine frequency steps (1 kHz, lower part).

Assuming  $f_{2\min} = 0$ , this inequality leads to

$$f_{\text{ofs}} > \frac{f_{1\max} - f_{1\min}}{2}$$

In our example,  $f_{\text{ofs}}$  would have to be chosen larger than 50 MHz. [Figure 6.12](#) is a possible implementation of the intended synthesizer.

The circuit consists of two synthesizers; hence, it is a dual-loop system. The frequency offset has been chosen to be  $f_{\text{ofs}} = 75$  MHz here. Let us consider the upper (coarse) synthesizer first. Without mixer Mix1, this loop would create frequencies in the range of 25 to 125 MHz, with a channel spacing of 1 MHz. Assume for the moment that  $f_2 = 0$ —that is, the frequency at the output of bandpass filter BP2 is exactly 75 MHz. Due to the mixer, VCO1 is forced to now create an output frequency that is higher by that offset—in other words, the output frequency range of VCO1 is now 100 to 200 MHz. The bandpass filter filters out the lower sideband only; hence, Mix1 operates as a frequency subtractor.

Now the fine frequency component must be added. With an offset of 75 MHz, the “fine” frequency  $f_2'$  must be in the range of 75 to 75.999 MHz with a channel spacing of 1 kHz. Basically, we could use a synthesizer with a reference frequency of 1 kHz and a scaling factor in the range of 75,000 to 75,999, but we remember that such a circuit would be slow and would have bad noise performance. Looking at the synthesizer in the lower part of the figure we recognize a synthesizer with reference 10 kHz instead of 1 kHz, creating a frequency range from 50 to 59.990 MHz with a channel spacing of 10 kHz. This synthesizer is a factor of 10 faster than a synthesizer using a 1 kHz reference. An external divide-by-10 counter scales down that range to between 5 and 5.999 MHz with a channel

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

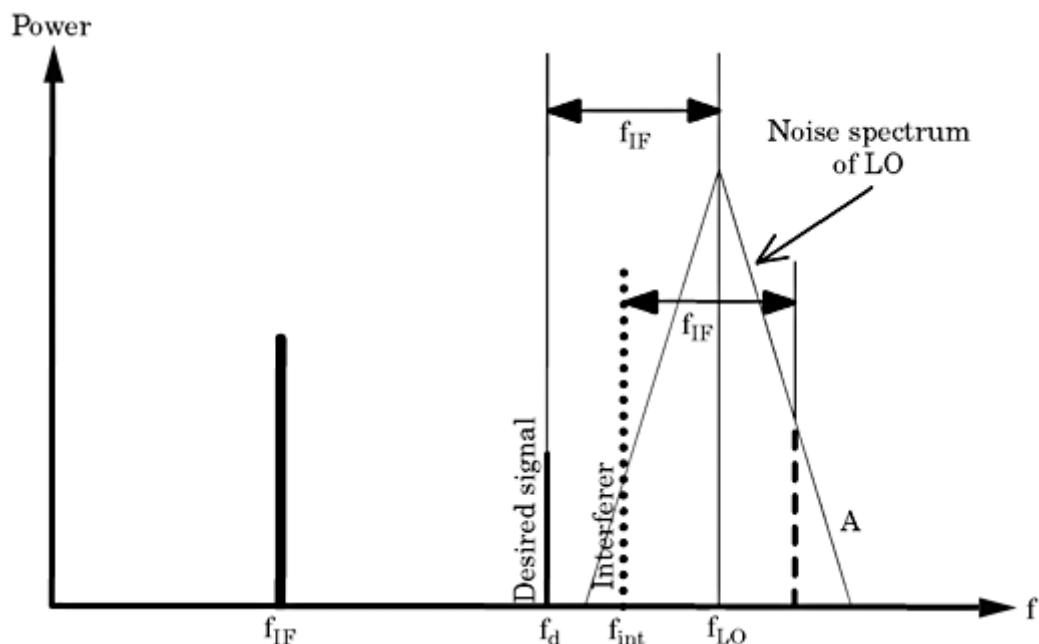
spacing 1 kHz. A second mixer Mix2 is used now to add a frequency offset of 70 MHz. Again Mix2 generates an upper and a lower sideband. The upper ranges from 75 to 75.999 MHz, the lower from 64.001 to 65 MHz. Because we use only the upper sideband, it is filtered out by a bandpass filter with center frequency of about 75.5 MHz. The one-sided bandwidth must be somewhat larger than about 0.5 MHz.

Multi-loop synthesizers are frequently found in signal generators, receivers, transmitters (for instance, short wave), and the like. Very sophisticated multi-loop designs have been described by Rohde.<sup>48, 49</sup>

## Phase Noise and Spurs in Integer Frequency Synthesizers

Having designed a PLL frequency synthesizer that uses a highly stable quartz-crystal reference oscillator, we may hope to get a nicely clean output signal with high frequency stability and no phase jitter. Mathematically, the spectrum of the synthesizer's output signal should consist of just one single line at the desired frequency. Unfortunately, reality shows another picture: when measuring the signal spectrum, we may observe quite a bit of phase jitter; moreover, we can detect a couple of sidebands (so called "spurs") around the desired center frequency.

We will demonstrate the adverse impacts of superimposed phase noise by the example of *reciprocal mixing* in mobile communications. Assume we want to detect a signal at a frequency  $f_d$ , which is the "desired signal" (refer to Fig. 6.13). Our receiver is supposed to use an intermediate frequency at  $f_{IF}$ . To get the IF signal we use a local oscillator (LO) generating the frequency  $f_{LO}$ , which is simply the sum  $f_d + f_{IF}$ . The phase noise spectrum is depicted by the triangular shaped curve symmetrical around frequency  $f_{LO}$ . The desired signal and LO output are



**Figure 6.13** Signal degradation at IF by reciprocal mixing.

---

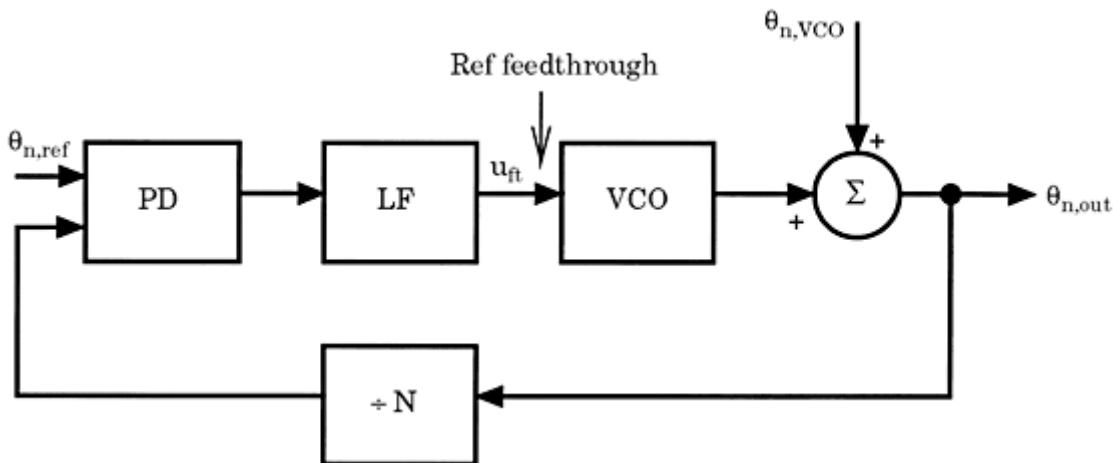
Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

fed to a mixer that creates the IF signal. Now assume that an interfering signal is present at frequency  $f_{\text{int}}$ , which is not far away from  $f_d$ , may be at a distance of only 200 kHz from  $f_d$ . This could be another communication signal that has perhaps a much greater amplitude than the desired signal. If the spectrum of the LO at frequency  $f_{\text{int}} + f_{\text{IF}}$  shows appreciable amplitude (as shown by line A in the figure), the interferer is also mixed down to the intermediate frequency. This phenomenon is called *reciprocal mixing*. When the interfering signal is 100 dB greater than that desired and the noise spectrum of the LO is “only” 100 dB below its carrier amplitude, these two signals produce the same mixer output power. In many mobile communications, the individual channels are spaced by 200 kHz. To avoid reciprocal mixing by an interferer whose frequency is immediately adjacent to the desired channel frequency, the noise spectrum of the LO must be markedly more than 100 dB below the carrier amplitude at a distance of 200 kHz from that carrier. Frequency synthesizers intended for such applications impose, therefore, very tight requirements on phase noise and spurs. It should be noted that in many applications the IF frequency  $f_{\text{IF}}$  is chosen 0 (zero IF systems).

In the following, we will investigate the sources of those undesired noise components. The mathematical analysis is quite cumbersome, but fortunately there are a number of models available that greatly simplify the analysis. Basically, each part of the synthesizer circuit can contribute to output phase jitter.<sup>48</sup> In the following, we will concentrate on the dominant ones. [Figure 6.14](#) shows a simplified model for the determination of output phase noise and spurious sidebands.

Three sources of phase jitter and spurs can be recognized:

- Phase jitter is created by the reference oscillator. Even the highest quality reference oscillator is not free from output phase jitter. This perturbation is denoted as  $\theta_{n,\text{ref}}$
- Phase jitter created by the VCO. Because the VCO is nothing else than an oscillator, it also will contribute to phase jitter. This perturbation is denoted as  $\theta_{n,\text{VCO}}$ .



**Figure 6.14** A model for analysis of output phase jitter  $\theta_{n,\text{out}}$  in a PLL frequency synthesizer.

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

■ As we have seen in [Sec. 2.4](#), the phase detector delivers an output signal that is proportional to the phase error  $\theta_e$  of the loop. This signal is a “quasi-DC” signal. In addition, the phase detector creates AC components at higher frequencies. Depending on the type of phase detector chosen, there may be an AC signal component at the reference frequency or at twice the reference frequency, which is the case for the EXOR. As we will see later in this section, in the case of the PFD with voltage output there can be even an AC component which is a *subharmonic* of the reference frequency. These unwanted AC components are partially suppressed by the loop filter, but the residual signal still will frequency-modulate the VCO output. When the frequency of that AC signal is equal to the reference frequency  $f_{\text{ref}}$ , for example, we will observe “spurs” at a distance of  $\pm f_{\text{ref}} \pm 2f_{\text{ref}} \dots$  from the carrier frequency  $f_0$ . The signal causing those spurs is denoted  $u_{\text{ft}}$  ( $\text{ft}$  = feedthrough) in [Fig. 6.14](#).

All these noise sources contribute to output phase jitter (labeled  $\theta_{n,\text{out}}$  in [Fig. 6.14](#)) and spurious sidebands. This all may sound disappointing, but when we succeed in quantifying the sources of trouble, we are in a position to minimize the undesired disturbances. In the following, we therefore analyze the three mentioned effects.

### Phase noise created by the reference oscillator

To analyze phase noise at the output of the reference oscillator, let us recall the noise theory presented in [Sec. 4.3](#). [Figure 4.4](#) has shown the relationships between signal power  $P_s$ , noise power  $P_n$ , and input phase jitter  $\theta_{n1}$ . In [Fig. 4.4a](#), the power spectral density (PSD) of noise power  $P_n$  was shown. This power spectrum has one-sided bandwidth  $B_f/2$  and is symmetrical around the center frequency  $f_0$  of the PLL. [Figure 4.4b](#) represented the PSD of input phase jitter  $\theta_{n1}$ . This spectrum also has the one-sided bandwidth  $B_f/2$  and is symmetrical around  $f=0$ . The input phase jitter has been shown to modulate the phase of the carrier frequency  $f_0$  (center frequency of the PLL). This was described by

$$u_1(t) = U_{10} \sin(\omega_1 t + \theta_{n1}(t)) \quad (6.4)$$

[cf. [Eq. \(2.2\)](#)]. For the following analysis we will use the phase noise model shown in [Fig. 6.15](#). The superposition of phase noise is accomplished by a phase modulator (PM). The phase noise  $\theta_{n1}(t)$  modulates the phase of the carrier having signal power  $P_s$ . The phase modulator is followed by a unity gain amplifier that is assumed for the moment to be noiseless.

When the input phase jitter contains a component at frequency  $f_m$ , we saw that this gives rise to two sidebands in the power spectrum  $P_n$  ([Fig. 4.4a](#)), one line being at frequency  $f_0 + f_m$ , the other at  $f_0 - f_m$ . Therefore, the spectra of input phase jitter and noise power are offset from each other by the carrier frequency  $f_0$ .

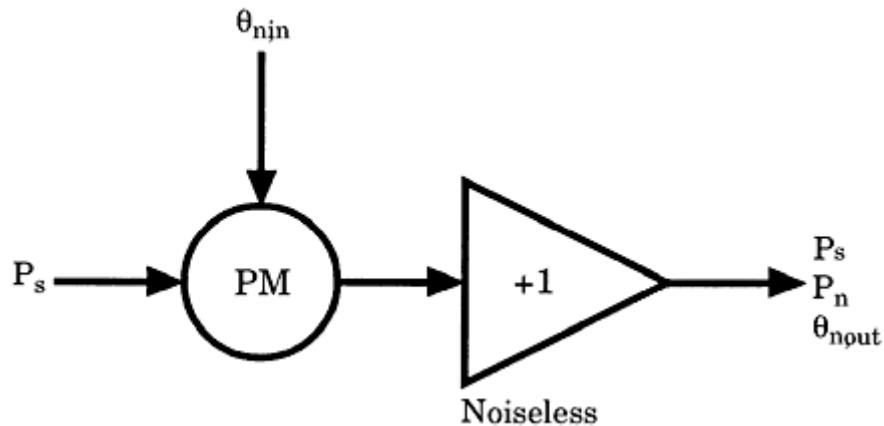
[Equation \(4.1\)](#) gave the relationship

$$(6.5)$$

$$\overline{\theta_{n1}^2} = \frac{P_n}{2P_s} \quad [\text{rad}^2]$$

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.



**Figure 6.15** Model demonstrating the effect of phase jitter onto a carrier signal having power  $P_s$ . The phase jitter creates phase modulation of the carrier (PM = phase modulator).

where  $\overline{\theta_{n1}^2}$  represents the mean square input phase jitter. This is identical with the shaded area under the squared phase spectrum  $|\Theta_{n1}|^2(f)$  in Fig. 4.4b.  $P_s$  is signal power (in W), and  $P_n$  is noise power (in W). The noise signal is denoted  $n(t)$ , and noise power is given by  $P_n = \overline{n(t)^2}$ . As we see from Fig. 4.4b,  $\overline{\theta_{n1}^2}$  is the mean square input phase jitter resulting from the *one-sided power spectrum*  $|\Theta_{n1}|^2(f)$ . But when there is a phase jitter component at frequency  $f_m$ , there is a correlated component at  $-f_m$  also, and consequently the overall mean square phase jitter becomes *twice as large*—that is

$$\overline{\theta_{n1}^2} = \frac{P_n}{P_s} \quad [\text{rad}^2] \quad (6.6)$$

From now on we specify with  $\overline{\theta_{n1}^2}$  the mean square phase jitter resulting from the *two-sided power density spectrum* of input phase jitter.

Equation (6.6) tells us how large the phase jitter will be when signal power  $P_s$  and noise power  $P_n$  are given. The unit of  $P_s$  and  $P_n$  is W, of course, and the unit  $\overline{\theta_{n1}^2}$  is rad<sup>2</sup>—meaning  $\overline{\theta_{n1}^2}$  is a mean square value. When analyzing noise performance in Sec. 4.3, we started from the premise that the noise spectrum would be “white”—in other words, that each frequency interval of width 1 Hz (within the input noise bandwidth  $B_i/2$ ) would contain the same power (in W/Hz). When dealing with phase jitter in oscillators, however, we will recognize that the corresponding noise spectra are not white at all, but will be highly nonlinear functions of frequency. It is not sufficient therefore to know the mean square phase jitter  $\overline{\theta_{n1}^2}$  alone, but the power spectral density of phase jitter  $\theta_{n1}(t)$  must also be known. To obtain the PSD of phase jitter, we will apply the PSD transform onto both sides of Eq. (6.6), which yields

$$S_{\theta} (f_m) = \frac{S_{nn}(f_m)}{P_s} \quad [\text{rad}^2/\text{Hz}] \quad (6.7)$$

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

where  $S_{\theta\theta}(f_m)$  is the “power spectral density of phase perturbation (jitter)”  $\theta_{n1}$  at (modulating) frequency  $f_m$ ; the unit is rad<sup>2</sup>/Hz.  $S_{nn}(f_m)$  is the power spectral density of the noise signal at a frequency that is displaced by the offset  $f_m$  from the carrier frequency; the unit is W/Hz. Finally, the unit of signal power  $P_s$  is W.

Next we are looking for a quantitative expression for the PSD of phase noise. Let's assume that the carrier power is  $P_s = 1$  mW and its frequency is  $f_0$ . The noise source is assumed to be thermal noise. Thermal noise has a power spectral density of  $kT$ —in other words

$$S_{nn}(f_m) = kT \quad [\text{W/Hz}] \quad (6.8)$$

with  $k$  = Boltzmann constant =  $1.4 \cdot 10^{-23}$  Ws/K

$T$  = absolute temperature in K (Kelvin)

At room temperature, we have  $T = 293$  K, and the noise spectral density becomes  $S_{nn}(f_m) = 0.41 \cdot 10^{-20}$  W/Hz. For  $S_{\theta\theta,\text{out}}(f_m)$ , we then get

$$S_{\theta,\text{out}}(f_m) = \frac{kT}{P_s} = 0.41 \times 10^{-17} \text{ rad}^2/\text{Hz}$$

Because this is an extremely small quantity,  $S_{\theta\theta,\text{out}}(f_m)$  is mostly expressed in a logarithmic scale—thus, in decibels. We therefore introduce a new variable  $S_{\theta\theta,\text{out}}(f_m)$  dB

$$S_{\theta,\text{out}}(f_m)_{\text{dB}} = 10 \log_{10} S_{\theta,\text{out}}(f_m) \quad [\text{dBc/Hz}] \quad (6.9)$$

The unit of  $S_{\theta\theta,\text{out}}(f_m)$  dB is dBc/Hz, and for the current example, the result is  $S_{\theta\theta,\text{out}}(f_m)$  dB =  $-174$  dBc/Hz. This tells us that the noise power contained within a bandwidth of 1 Hz (located at a frequency that is offset by  $f_m$  from the carrier frequency) is 174 dB below the power of the carrier. The letter “c” in the unit dBc signifies that  $S_{\theta\theta,\text{out}}(f_m)$  dB stands for “noise power referred to carrier power.” This value  $S_{\theta\theta,\text{out}}(f_m)$  dB =  $-174$  dBc/Hz is the absolute best result we could get from an amplifier, since thermal noise is always present and real amplifiers create additional noise. To get an idea of phase jitter to expect in this example, we compute its rms value—thus, the square root of  $S_{\theta\theta,\text{out}}(f_m)$ . This yields

$$\theta_{n,rms} = \sqrt{S_{\theta,\text{out}}(f_m)} = \sqrt{0.41 \cdot 10^{-17}} = 2.02 \times 10^{-9} \text{ rad/Hz}$$

**Note** The unit dBc/Hz is widely used in textbooks, data sheets, and application notes on PLL frequency synthesizers. From a mathematical point of view, however, this unit is not entirely correct for the reasons explained in the following. As we know,  $S_{\theta\theta}(f_m)$  as defined in Eq. (6.7) represents a ratio of noise power to carrier power. If total

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

noise power is  $P_n$  and carrier power is  $P_s$ , a noise-to-carrier ratio (NCR) could be defined by

$$\text{NCR} = \frac{P_n}{P_s} \quad [\text{W/W}]$$

Note that the unit of NCR (W/W) is dimensionless. Usually one defines another noise-to-carrier ratio  $\text{NCR}_{\text{dB}}$  expressed in decibels:

$$\text{NCR}_{\text{dB}} = 10 \log_{10}(\text{NCR}) = 10 \log_{10} \frac{P_n}{P_s}$$

Checking [Eq. \(6.7\)](#) once again, we recognize that the ratio  $S_{nn}(f_m)/P_s$  is not dimensionless, since  $S_{nn}(f_m)$  does not stand for *power*, but for *power density*, whose unit is W/Hz. The variable  $P_s$ , however, has the unit W, hence the ratio  $S_{nn}(f_m)/P_s$  has the unit  $\text{Hz}^{-1}$ . Mathematically, it is not correct to build the logarithm of a ratio that is not dimensionless. [Try to find out what  $\log(\text{Hz}^{-1})$  is.] We can circumvent that dilemma by introducing new variables  $S_{nn}^*(f_m)$  and  $S_{\theta\theta}^*(f_m)$  as follows:

$$S_{nn}^*(f_m) = S_{nn}(f_m) \cdot B \text{ where } B = 1 \text{ Hz} \quad [\text{W}]$$

$$S_{\theta\theta}^*(f_m) = S_{\theta\theta}(f_m) \cdot B \quad [\text{rad}^2]$$

$B$  stands for bandwidth and is set  $B = 1$  Hz. Due to multiplication with  $B$ , the unit of  $S_{nn}^*(f_m)$  becomes W (and not W/Hz). The new variable  $S_{nn}^*(f_m)$  now signifies *noise power* (not *power density*) within a bandwidth  $B = 1$  Hz, located at an offset  $f_m$  from the carrier frequency. The new variable  $S_{\theta\theta\pi}^*(f_m)$  is defined by

$$S_{\theta\theta}^*(f_m) = \frac{S_{nn}^*(f_m)}{P_s}$$

and is now the ratio of two power quantities. Its unit is therefore  $\text{rad}^2$  and not  $\text{rad}^2/\text{Hz}$ .  $S_{\theta\theta}^*(f_m)$  therefore no longer represents the power density of phase perturbations, but rather the mean square value of phase perturbation  $\overline{\theta_{n1}^2}$  whose spectrum ranges from  $f_m < f < f_m + 1$ . It is now mathematically correct to build a logarithmic quantity from  $S_{\theta\theta\pi}^*(f_m)$  by setting

$$S_{\theta\theta}^*(f_m)_{\text{dB}} = 10 \log_{10} S_{\theta\theta}^*(f_m) \quad [\text{dBc}]$$

The unit of  $S_{\theta\theta}^*(f_m)_{\text{dB}}$  now becomes dBc and not dBc/Hz. Let's do a numerical example. Assume that the carrier power is  $P_s = 1$  mW, and that the noise power density  $S_{nn}(f_m)$  is  $10^{-15}$  W/Hz at an offset  $f_m = 10$  kHz from the carrier frequency. The noise power  $S_{nn}^*(f_m)$  within a bandwidth of 1 Hz at  $f_m = 10$  kHz is then given by

$$S_{nn}*(10000) = 10^{-15} \frac{\text{W}}{\text{Hz}} \cdot 1 \text{ Hz} = 10^{-15} \text{ W}$$

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

This is the noise power within the frequency interval from 10,000 to 10,001 Hz relative to the carrier frequency. Next  $S_{\theta\theta}^*(f_m)$  is to be computed. We get

$$S_{\theta}^*(f_m) = \frac{P_{nn}^*(f_m)}{P_s} = \frac{10^{-15} \text{ W}}{10^{-3} \text{ W}} = 10^{-12}$$

Hence,  $S_{\theta\theta}^*(f_m)_{\text{dB}}$  becomes

$$S_{\theta}^*(f_m)_{\text{dB}} = 10 \log_{10} 10^{-12} = 120 \text{ dBc}$$

We conclude that the noise power within a bandwidth of 1 Hz at offset frequency  $f_m = 10$  kHz is 120 dB below carrier power. Furthermore, we see that the mean square value of phase perturbation within a bandwidth of 1 Hz at modulating frequency  $f_m = 10$  kHz is  $10^{-12} \text{ rad}^2$ . For the rms value of phase perturbation in that frequency interval we get

$$\theta_{n1,rms} = \sqrt{\overline{\theta_n^2}} = \sqrt{S_{\theta}^*(f_m)} = \sqrt{10^{-12}} = 10^{-6} \text{ rad}$$

To avoid confusion with the standards used in practically all books and papers on frequency synthesizers, we will nevertheless use the familiar unit dBc/Hz for power density of phase jitter, although this unit has been shown to be somewhat incorrect.

As mentioned earlier, real amplifiers add further noise. Noise performance of amplifiers is specified by noise figure  $F$  which is defined by

$$F = \frac{P_{s,\text{out}}/P_{n,\text{out}}}{P_{s,\text{in}}/P_{n,\text{in}}} \quad (6.10)$$

in other words,  $F$  is the ratio of signal-to-noise at the output to signal-to-noise at the input. For a real amplifier  $S_{\theta\theta,\text{out}}(f_m)$  becomes larger than  $S_{\theta\theta,\text{in}}(f_m)$  by factor  $F$ ; thus, we now have

$$S_{\theta,\text{out}}(f_m) = F \cdot S_{\theta,\text{in}}(f_m) = \frac{kTF}{P_s}$$

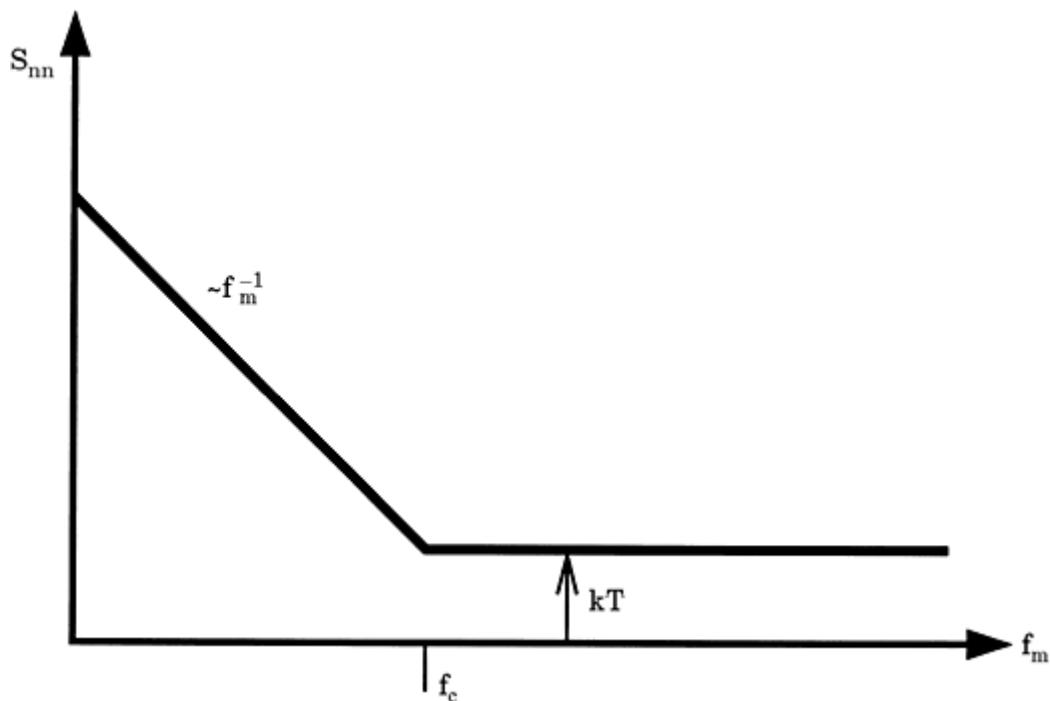
When we assume  $F = 6$  dB,  $S_{\theta\theta,\text{out}}(f_m)_{\text{dB}}$  increases by 6 dB, thus

$$S_{\theta\theta,\text{out}}(f_m)_{\text{dB}} = -174 + 6 = -168 \text{ dBc/Hz}$$

In real amplifiers, we are not only confronted with thermal noise, but there is always another noise source called *flicker noise*. Flicker noise is also referred to as 1/f noise, because the power spectral density of flicker noise varies with  $f_m^{-1}$  amplifier, however, at frequencies below a corner frequency denoted as  $f_c$  [cf. Fig. 6.16]. For operational amplifiers, for example,  $f_c$  can be on the order of 10 Hz to some kHz. In oscillators,  $f_c$  can easily extend into the MHz region.

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.



**Figure 6.16** The power spectral density of input phase jitter  $\theta_{n,\text{in}}$ . Noise below the corner frequency  $f_c$  is called flicker noise. Noise above  $f_c$  is thermal noise.

When flicker noise is present, we get for the PSD of the noise signal

$$S_{nn}(f_m) = kTF \left( 1 + \frac{f_c}{f_m} \right) [\text{W/Hz}] \quad (6.11)$$

Now we get for the PSD of phase jitter the expression

$$S_{\theta,\text{out}}(f_m) = \frac{kTF}{P_s} \left( 1 + \frac{f_c}{f_m} \right) [\text{rad}^2/\text{Hz}] \quad (6.12)$$

Still assuming  $F = 6 \text{ dB}$  we get for a modulating frequency  $f_m = f_c/10$  the value

$$S_{\theta,\text{out}}(f_m)_{\text{dB}} = -174 + 6 + 10 = -158 \text{ dBc/Hz}$$

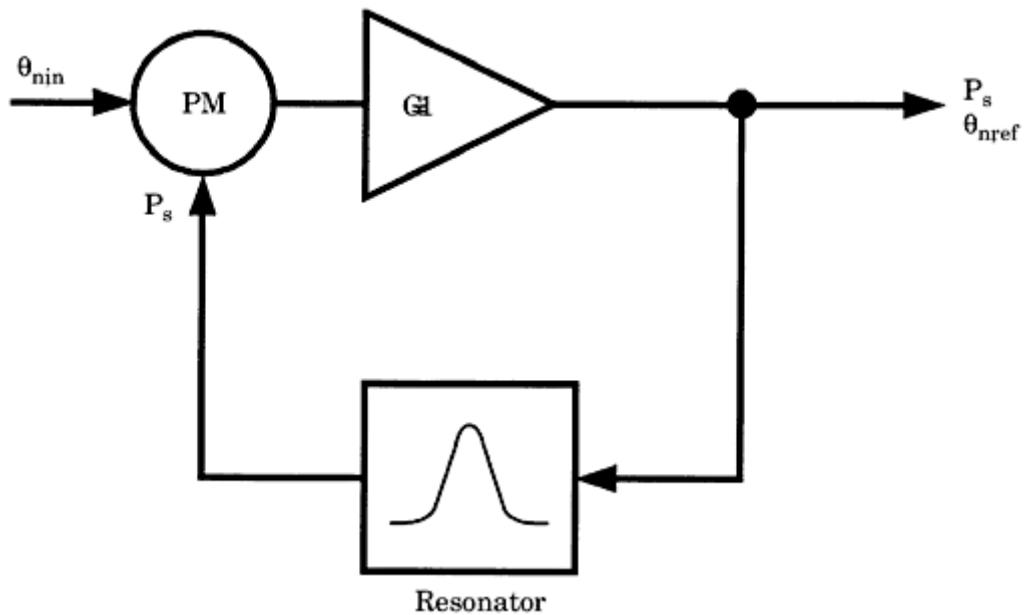
At a modulating frequency  $f_m = f_c/100$ , this would increase to  $-148 \text{ dBc/Hz}$ .

We considered hitherto the very simple case where a noise source was modulating a carrier signal at the input of a noiseless or of a real amplifier. Next, we have to examine the situation where the noise source is fed to the input of an oscillator. The noise spectrum observed at the oscillator output deviates considerably from that of a simple amplifier because the oscillator has a frequency-dependent loop gain. In the vicinity of the resonant frequency the loop gain of the oscillator is extremely high, hence the noise will be heavily amplified at those frequencies.

To get the PSD of phase noise at the output of an oscillator we will

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.



**Figure 6.17** Model for the analysis of phase jitter in an oscillator (PM = phase modulator).

use the model of Fig. 6.17. The noise signal fed to the input of the oscillator is denoted as  $\theta_{n,\text{in}}(t)$  and its PSD is  $S_{\theta\theta,\text{in}}(f_m)$ . The phase noise at the output of the oscillator is denoted as  $\theta_{n,\text{ref}}(t)$  and its PSD is  $S_{\theta\theta,\text{ref}}(f_m)$ .

The oscillator is represented by a closed loop having positive feedback. In the forward path, we have an amplifier with power gain  $G = 1$ . A resonator is placed in the feedback path. Due to the feedback path,  $S_{\theta\theta,\text{ref}}(f_m)$  is no longer identical with  $S_{\theta\theta,\text{in}}(f_m)$  as in the model of Fig. 6.15, but is given by

$$S_{\theta,\text{ref}}(f_m) = S_{\theta,\text{in}}(f_m) \cdot |G_n(f_m)|^2 \quad [\text{rad}^2/\text{Hz}] \quad (6.13)$$

Here,  $G_n(f_m)$  is the closed-loop gain of the oscillator. As shown by Rohde,<sup>48</sup> the squared closed-loop gain is given by

$$|G_n(f)|^2 = 1 + \frac{1}{f_m^2} \left( \frac{f_0}{2Q} \right)^2 \quad (6.14)$$

where  $f_0$  = resonant frequency of the oscillator

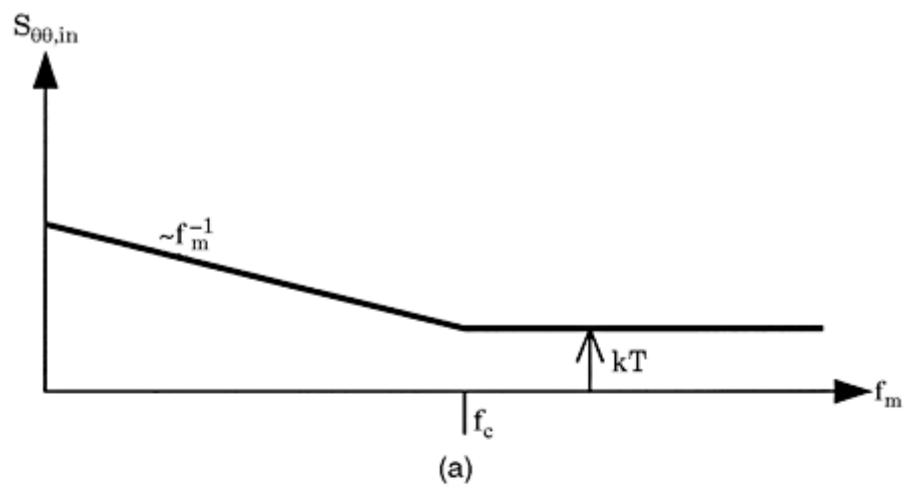
$f_m$  = frequency offset from resonant frequency (i.e.  $f_m = f - f_0$ )

$Q$  = quality factor of the resonator

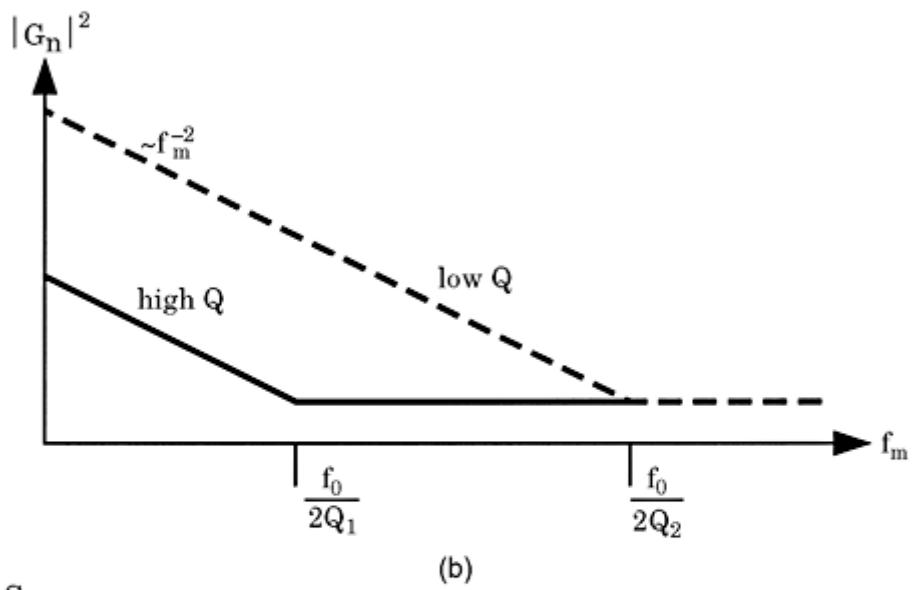
At frequencies far away from the carrier frequency  $f_0$ , the closed-loop gain is unity. The term  $f_0/2Q$  is the one-sided bandwidth of the resonator. Below the corner frequency  $f_0/2Q$ , the

closed-loop gain increases with  $f_m^{-2}$ . This is shown in Fig. 6.18b for two cases, a low  $Q$  ( $Q = Q_2$ ) and a high  $Q$  ( $Q = Q_1$ ). Figure 6.18a sketches once more the power density spectrum of the equivalent input noise. Besides thermal noise  $kT$ , there is flicker noise below corner frequency  $f_c$ . The

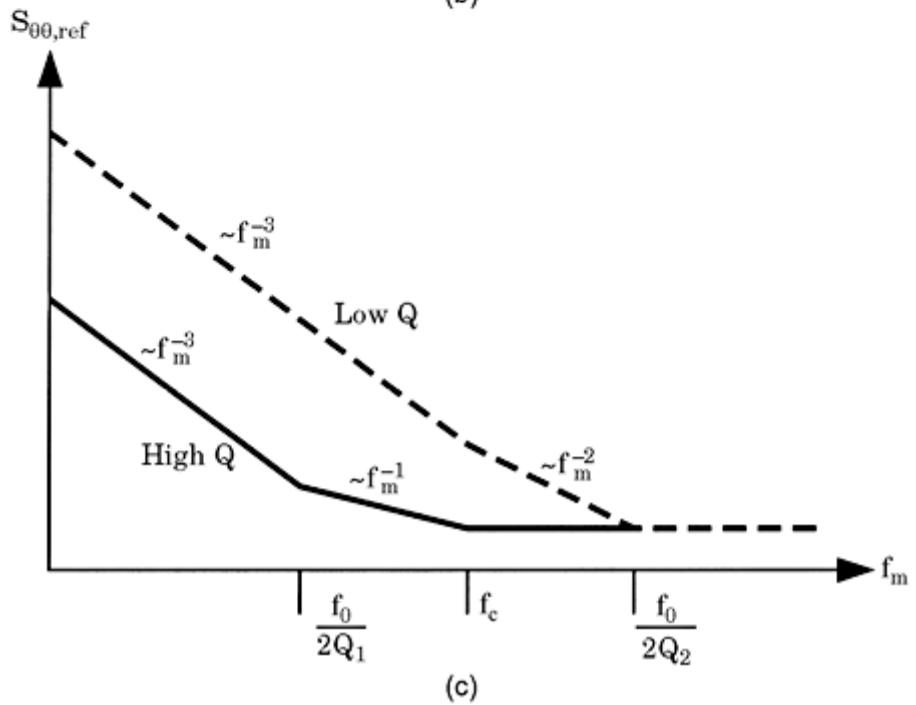




(a)



(b)



(c)

**Figure 6.18** Power density spectra of phase jitter in an oscillator. (a) Power density spectrum of the noise applied to the oscillator input. (b) Squared closed-loop gain  $|G_n(f_m)|^2$  of the oscillator. The solid line represents an oscillator having a resonator with high  $Q$ , while the dashed line represents an oscillator with a low  $Q$  resonator ( $Q_1 > Q_2$ ). (c) The power density spectrum of the oscillator's output phase jitter  $S_{\theta\theta,\text{ref}}(f_m)$ .

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

two values for  $Q$  have been chosen such that the corner frequency  $f_0/2Q_1$  is below  $f_c$ , and the corner frequency  $f_0/2Q_2$  is above. Let's first discuss the high- $Q$  case. The power density spectrum of the output phase jitter is obtained by multiplying the appropriate curves in Fig. 6.18a and b. The result is plotted in Fig. 6.18c. For high  $Q$ ,  $S_{\theta\theta,\text{ref}}(f_m)$  is constant versus frequency for  $f < f_c$ . Under ideal conditions (that is, for a noise figure of  $F = 1$ ),  $S_{\theta\theta,\text{ref}}(f_m)$  dB approaches  $-174$  dB for a signal power of  $1$  mW ( $0$  dBm). Below  $f_c$ ,  $S_{\theta\theta,\text{ref}}(f_m)$  increases with  $f_m^{-1}$ , and below corner frequency  $f_0/2Q_1$ , it increases with  $f_m^{-3}$ .

In the low  $Q$  case,  $S_{\theta\theta,\text{ref}}$  is constant above  $f_0/2Q_2$ . In the range between  $f_c$  and  $f_0/2Q_1$ , it increases with  $f_m^{-2}$ , and below  $f_c$  it increases with  $f_m^{-3}$ . What we see drastically from Fig. 6.18, the resonator  $Q$  has a tremendous effect on the oscillator phase jitter. The higher the  $Q$ , the lower the phase jitter. The spectral density of oscillator phase jitter is greatest near the resonance frequency; this could be expected because the noise gain of the oscillator is maximum at resonance frequency  $f_0$ .

Going back to the synthesizer noise model in Fig. 6.14, we now know how large the phase jitter  $\theta_{n,\text{ref}}$  could be. But what is the impact of that phase jitter on the synthesizer output  $\theta_{n,\text{out}}$ ? We know from Sec. 3.3.1 how the PLL reacts on phase signals  $\pi_1(t)$  applied to the reference input. To compute the output phase signal  $\theta_2'(t)$ , we found

$$H(s) = \frac{\Theta_2'(s)}{\Theta_1(s)} \quad (6.15)$$

[cf. Eq. (3.7)]. In the case of the frequency synthesizer, however, we are not interested in phase jitter  $\theta_2'(t)$  (which appears at the output of the divide-by- $N$  counter), but rather in the phase jitter  $\theta_2(t)$  at the *output of the VCO*. As is easily seen from the block diagram in Fig. 2.1,  $\theta_2$  and  $\theta_2'$  are related by  $\theta_2 = \theta_2' \cdot N$ . For the power spectral density of the phase jitter at the output of the VCO, we consequently have

$$S_{\theta,\text{out}}(f_m) = |H(f_m)|^2 \cdot N^2 \cdot S_{\theta,\text{ref}}(f_m) \quad [\text{rad}^2/\text{Hz}] \quad (6.16)$$

Here again,  $f_m$  is the frequency offset from the carrier frequency  $f_0$ . If  $f_m$  lies within the passband of the PLL (given by  $B_L$  or  $f_{3db}$ ),  $H(f_m) \approx 1$ —thus, the synthesizer multiplies the reference phase jitter by  $N^2$ . In the stopband, however, phase jitter is attenuated; the attenuation depends on the gain roll-off of  $H(f)$  at higher frequencies, which will be treated in greater detail in Chap. 9.

In most cases, the reference oscillator is not directly connected to the phase detector input, but rather the frequency created by the reference oscillator is scaled down by a reference divider, as shown, for example, in Fig. 6.2. When the reference divider scales down the frequency by  $R$ , it also attenuates the oscillator phase jitter by  $R$ , which decreases the PSD of phase jitter by  $R^2$ . When both  $\div R$  and  $\div N$  dividers are used, the PSD of the VCO output becomes

$$S_{\theta, \text{out}}(f_m) = |H(f_m)|^2 \cdot \frac{N^2}{R^2} \cdot S_{\theta, \text{ref}}(f_m) \quad [\text{rad}^2/\text{Hz}] \quad (6.17)$$

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

We note that the phase frequency response  $H(f)$  acts as a lowpass filter onto the phase jitter caused by the reference oscillator. To minimize VCO output jitter, the loop bandwidth ( $B_L$ ) must be made as small as possible. But this bandwidth cannot be chosen arbitrarily low, because a low  $B_L$  also means a low natural frequency  $\omega_n$ —in other words, a large lock-in time.

## Phase noise created by the VCO

In [Sec. 6.7.1](#), we investigated the phase jitter introduced by the reference oscillator. Checking the phase jitter model of [Fig. 6.14](#) once again, we recognize that the VCO is nothing else than an oscillator, too, hence it will introduce additional phase jitter into the loop. This phase noise (denoted  $\theta_{n,VCO}$  in the block diagram) enters the loop at the input of the summator labeled  $\Sigma$ . If there were no feedback, phase noise  $\theta_{n,VCO}$  would simply add to the output phase noise  $\theta_{n,out}$ . Because we have a feedback path, we must compute the closed-loop gain  $G_n$  from the insertion point of the VCO phase noise  $\theta_{n,VCO}$  to the output terminal  $\theta_{n,out}$ . For  $G_n$ , we obtain the very simple result

$$G_n(f) = H_e(f) \quad (6.18)$$

in other words, it is identical with the *error frequency response* as defined in [Eq. \(3.20\)](#). As can be seen from the Bode diagram for  $H_e(\omega)$  in [Fig. 3.3](#),  $H_e(\omega)$  is a *highpass* function. The PSD of phase noise contributed by the VCO is given by

$$S_{\theta,out}(f_m) = |H_e(f_m)|^2 \cdot S_{\theta,VCO}(f_m) \quad [\text{rad}^2/\text{Hz}] \quad (6.19)$$

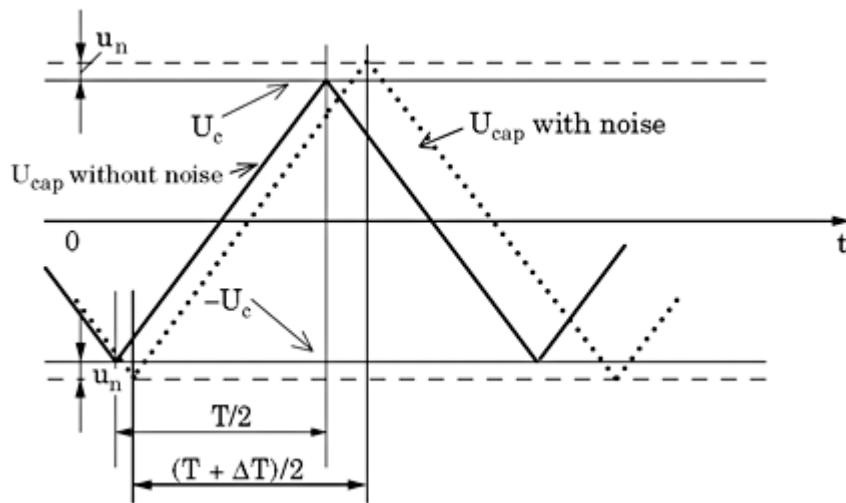
with  $S_{\theta,VCO}(f_m)$  = PSD of phase jitter introduced by the VCO,  $S_{\theta,out}(f_m)$  = PSD of output phase jitter due to VCO phase jitter, and  $f_m$  = frequency offset from center frequency  $f_0$ . The bandwidth of highpass function  $H_e(f)$  is the same as the bandwidth of the lowpass function  $H(f)$ , because  $H_e(f) = 1 - H(f)$  [c.f. [Eq. \(3.9\)](#)]. If we assume that most of the VCO phase noise is flicker noise, most of its power is concentrated around the center frequency  $f_0$ . To minimize output phase jitter, it would be optimum to make the PLL bandwidth as large as possible. This is in clear contradiction to the conclusion in [Sec. 6.7.1](#) where we stated that the loop bandwidth should be made small in order to reduce reference phase jitter.

As we have seen in [Sec. 2.6](#), there are two entirely different types of VCOs in use: *relaxation oscillators* and *resonant oscillators*. For the latter, the PSD of oscillator phase noise has been analyzed in [Sec. 6.7.1](#), hence we can adopt these results to compute the overall output phase noise  $S_{\theta,out}(f_m)$  immediately.

For relaxation oscillators, however, no phase noise model is known at present. The author tried therefore to develop such a model. This will be presented next.

**Noise model for relaxation type VCOs.** The operating principle of the relaxation oscillator has been explained in [Sec. 2.6.1](#) (refer to [Fig. 2.23](#)). It was shown that the voltage  $U_{cap}$  across capacitor  $C$  has a triangular waveform, as shown by the solid curve in [Fig. 6.19](#). The

waveform is symmetrical around DC and oscillates



**Figure 6.19** Voltage  $U_{\text{cap}}$  across capacitor  $C$  in the relaxation oscillator as shown in Fig. 2.23. Plotted is the triangular waveform of  $U_{\text{cap}}$  without comparator noise  $u_n$  (solid curve) and with comparator noise  $u_n$  (dotted line).

between the levels  $U_c$  and  $-U_c$ , where  $U_c$  is the threshold voltage of the two comparators in Fig. 2.23. This is depicted by the solid line in Fig. 6.19. The duration of one cycle of the capacitor waveform is denoted  $T$ . Now assume that there is voltage noise  $u_n$  superimposed to the comparator threshold  $U_c$ . Let us furthermore assume for the moment that this noise signal varies slowly during one cycle of the triangular wave, so we can draw it as a horizontal line. With added noise, the triangular waveform varies from  $-U_c - u_n$  to  $U_c + u_n$ . The duration of one cycle now increases to  $T + \Delta T$ . We will now calculate  $T$  and  $\Delta T$ . When the current charging the capacitor is  $I_0$ , the cycle time  $T$  becomes

$$T = \frac{4U_c}{I_0 C}$$

and the frequency of the triangular waveform becomes

$$f_0 = \frac{1}{T} = \frac{I_0 C}{4U_c}$$

With added noise, the frequency becomes dependent on  $u_n$

$$f = \frac{1}{T + \Delta T} = \frac{I_0 C}{4(U_c + u_n)} = \frac{f_0}{1 + \frac{u_n}{U_c}}$$

Because  $u_n/U_c \ll 1$ , this can be simplified to

$$f \approx f_0 \left( 1 - \frac{u_n}{U_c} \right) \quad (6.20)$$

Let's assume for the moment that  $u_n$  is a sine wave with radian frequency  $\omega_m$  and amplitude  $\hat{U}_n$

$$u_n(t) = \hat{U}_n \sin(\omega_m t) \quad (6.21)$$

Inserting Eq. (6.21) into Eq. (6.20) yields

$$\omega(t) = \omega_0 \left( 1 - \frac{\hat{U}_n \sin \omega_m t}{U_c} \right) \quad (6.22)$$

with  $\omega = 2\pi f$  and  $\omega_0 = 2\pi f_0$ .  $\omega$  is the radian frequency of a modulated carrier with carrier radian frequency  $\omega_0$ . The peak frequency deviation is  $\frac{\hat{U}_n}{U_c}$ , so we can state that the signal  $u_n(t)$  modulates the frequency of the carrier. Knowing the radian frequency of the frequency modulated signal, we can compute its phase  $\varphi(t)$  by integrating  $\omega$  over time  $t$ . This leads to

$$\varphi(t) = \omega_0 t + \frac{\hat{U}_n}{U_c} \frac{\omega_0 \cos \omega_m t}{\omega_m} \quad (6.23)$$

Note that  $\varphi(t)$  is the phase of a frequency modulated triangular wave. Because the harmonics of the triangular signal are of no importance since they are suppressed almost entirely by the loop filter, it is sufficient to account for the fundamental only. The fundamental of the capacitor voltage (denoted  $U_{\text{cap},1}$ ) is a sine wave with amplitude  $U_0$  whose phase is given by  $\varphi(t)$ , thus

$$U_{\text{cap},1}(t) = U_0 \sin \left( \omega_0 t + \frac{\hat{U}_n}{U_c} \frac{\omega_0 \cos \omega_m t}{\omega_m} \right) \quad (6.24)$$

When computing the Fourier series of a symmetrical triangular signal, the fundamental  $U_0$  can be shown to be  $U_0 = \frac{4}{\pi^2} U_c \approx 0.4 U_c$ .

Applying the addition theorem of trigonometric functions to Eq. (6.24) gives

$$\begin{aligned} U_{\text{cap},1}(t) &= U_0 \left\{ \sin \omega_0 t \cos \left[ \frac{\hat{U}_n}{U_c} \frac{\omega_0 \cos \omega_m t}{\omega_m} \right] \right. \\ &\quad \left. + \cos \omega_0 t \sin \left[ \frac{\hat{U}_n}{U_c} \frac{\omega_0 \cos \omega_m t}{\omega_m} \right] \right\} \end{aligned} \quad (6.25)$$

Now we observe that the argument of the cosine function in square brackets is much less

than 1, so we replace the cosine term by 1. For the same reason, we can replace the sine term by its argument, which yields

$$U_{\text{cap},1}(t) \approx U_0 \left[ \sin \omega_0 t + \cos \omega_0 t \frac{\hat{U}_n}{U_c} \frac{\omega_0 \cos \omega_m t}{\omega_m} \right] \quad (6.26)$$

By applying the addition theorem of trigonometric functions once again, we obtain

$$U_{\text{cap},1}(t) \approx U_0 \left[ \sin \omega_0 t + \frac{\hat{U}_n \omega_0}{2U_c \omega_m} \cdot \cos(\omega_0 + \omega_m)t + \frac{\hat{U}_n \omega_0}{2U_c \omega_m} \cdot \cos(\omega_0 - \omega_m)t \right] \quad (6.27)$$

Clearly the capacitor voltage consists of three terms; the first one is the “carrier” at radian frequency  $\omega_0$ , and the other terms represent upper and lower sidebands at radian frequencies  $\omega_0 + \omega_m$  and  $\omega_0 - \omega_m$ , respectively. These sidebands are created by the noise signal  $u_n$  added to the comparator threshold  $U_c$ . The side-bands create a phase perturbation of the signal across the capacitor. Applying the same theory we used to compute the phase noise of the reference oscillator [cf. [Sec. 6.7.1](#) and [Eq. \(6.7\)](#)], the phase perturbation can be computed from

$$S_{\theta, \text{vco}} = \frac{\text{sideband power}}{\text{carrier power}}$$

The amplitude of one sideband is  $\frac{U_0 \hat{U}_n(f_m) \omega_0}{2U_c \omega_m}$  or  $\frac{U_0 \sqrt{2} U_{n,rms}(f_m) \omega_0}{2U_c \omega_m}$ , where  $U_{n,rms}$  is the rms value of the noise signal. The rms value of one sideband is therefore  $\frac{U_0 U_{n,rms}(f_m) \omega_0}{2U_c \omega_m}$ , and

its power is the rms value squared—in other words,  $\frac{U_0^2 U_{n,rms}^2(f_m) \omega_0^2}{4U_c^2 \omega_m^2}$ . Because we have two sidebands, the total noise power is twice as much:  $\frac{U_0^2 U_{n,rms}^2(f_m) \omega_0^2}{2U_c^2 \omega_m^2}$ . Because the carrier power is  $\frac{U_0^2}{2}$ , the phase perturbation becomes

$$S_{\theta, \text{vco}}(f_m) = \frac{U_{n,rms}^2(f_m) f_0^2}{U_c^2 f_m^2} \quad (6.28)$$

This is the phase perturbation of the capacitor voltage  $U_{\text{cap},1}$  caused by a noise signal having frequency  $f_m$ . Noise signals are normally broadband signals, thus we define by  $U_{n,rms}^2(f_m)$  the power spectral density of the noise signal measured in W/Hz (we assume, as usual, that the signal is applied to a load resistor of  $1 \Omega$ ). Then  $U_{n,rms}^2(f_m) \cdot B$ ,  $B = 1 \text{ Hz}$  is the noise power in a band that is  $1 \text{ Hz}$  wide at frequency  $f_m$ . We now are in the position to

compute the power density spectrum of the phase perturbation from

$$S_{\theta, \text{vco}}(f_m) = \frac{U_{n,rms}^2(f_m) f_0^2 B}{U_c^2 f_m^2} \quad [\text{rad}^2/\text{Hz}] \quad (6.29)$$

In noise calculations, it is customary to use logarithmic units (dB), so we define

$$S_{\theta, \text{vco}}(f_m)_{\text{dB}} = 10 \log_{10} \frac{U_{n,rms}^2(f_m) f_0^2 B}{U_c^2 f_m^2} \quad [\text{dB}/\text{Hz}] \quad (6.30)$$

We assumed hitherto that  $U_{n,rms}^2(f_m)$  is constant over frequency, hence it has a white spectrum. In reality, however, we must account for flicker noise, too. In analogy with [Sec. 6.7.1](#) and [Eq. \(6.11\)](#), we can define the noise spectral density as

$$U_{n,rms,tot}^2(f_m) = U_{n,rms}^2(f_m) \left(1 + \frac{f_c}{f_m}\right) \quad [\text{W/Hz}] \quad (6.31)$$

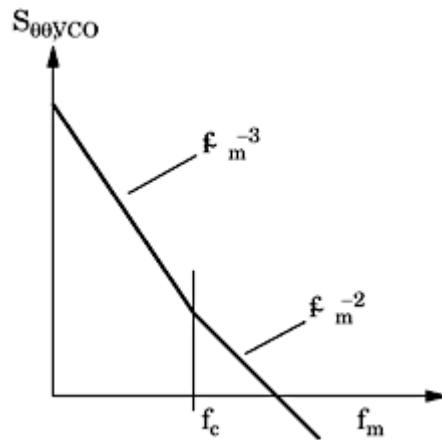
where  $U_{n,rms}^2(f_m)$  is a constant and  $f_c$  is the corner frequency of flicker noise. For the phase perturbation of the VCO output signal, we can therefore write

$$S_{\theta,\text{VCO}}(f_m)_{\text{dB}} = 10 \log_{10} \frac{U_{n,rms}^2(f_m) f_0^2 B}{U_c^2 f_m^2} \left(1 + \frac{f_c}{f_m}\right) \quad [\text{dB/Hz}] \quad (6.32)$$

The phase perturbation  $S_{\theta,\text{VCO}}$  is plotted versus frequency  $f_m$  in [Fig. 6.20](#). Note that the VCO operates at its center frequency  $f_0$ , and that  $f_m$  is an offset from the center frequency. As shown earlier, a noise component  $u_n$  at frequency  $f_m$  generates sidebands at frequencies  $f_0 + f_m$  and  $f_0 - f_m$ . The noise power density is largest in the vicinity of the center frequency. It decays with  $f_m^{-3}$  for frequency offsets less than the flicker corner frequency  $f_c$  and with  $f_m^{-2}$  above  $f_c$ .

There is still another way to compute noise in relaxation VCO: One could suppose that the current charging the capacitor  $C$  (cf. [Fig. 2.23](#)) is not constant but contains a noise component  $i_n(f_m)$ . The noise current then also frequency modulates the triangular signal across capacitor  $C$ . When performing the corresponding analysis, one gets a similar result. Assuming there exists flicker noise, the power density spectrum of the phase perturbation looks like that in [Fig. 6.20](#).

Unfortunately, the manufacturers of relaxation type VCO do not specify noise parameters. Thus, we don't have any idea about the size of variables like  $U_{n,rms}^2(f_m)$  or corner frequency  $f_c$ . Theoretically, such noise parameters can be measured, but the required instrumentation is quite complex. Measurement procedures for noise spectra are presented, for example, by Rohde.<sup>11</sup>



**Figure 6.20** Phase perturbation  $S_{\theta\theta,\text{VCO}}$  versus modulating frequency  $f_m$ .

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

**Noise model for resonant type VCOs.** The situation is much better for resonant VCOs because we can use the noise model we already developed for the reference oscillator (see Sec. 6.7.1). The final results have been given in Eqs. (6.11) through (6.14).

**Contribution of VCO phase noise to phase noise at PLL output.** Knowing the noise models of both resonant and relaxation type VCOs, we can now compute the contribution of VCO phase noise to total phase noise at the output of the PLL. For this analysis, the noise model in Fig. 6.14 is used again. Given the phase perturbation  $S_{\theta\theta,VCO}$  created by the VCO, the resulting phase perturbation  $S_{\theta\theta,out}$  at the PLL output becomes

$$S_{\theta,\text{out}}(f_m) = H_e^2(f_m) \cdot S_{\theta,\text{VCO}}(f_m) \quad [\text{rad}^2/\text{Hz}] \quad (6.33)$$

The phase noise generated by the VCO is therefore “weighted” by the error-transfer function  $H_e(s)$ , as defined in Eq. (3.8). As we know,  $H_e(s)$  is a highpass function. The low frequencies of phase noise created by the VCO are therefore almost fully suppressed. The high frequencies come through unattenuated because the magnitude of  $H_e(s)$  becomes 1 at higher frequencies. This phenomenon is easily explained by checking the noise model Fig. 6.14 once again. At low frequencies, the phase noise  $\theta_{n,VCO}$  is almost completely compensated for by the loop, because at low frequencies the loop gain is very high. This phase noise acts like a phase signal  $\pi_1$  applied at the reference input—that is, at the upper input of the phase detector. At higher frequencies, the loop gain decreases toward zero, and the phase noise  $\theta_{n,VCO}$  is no longer attenuated.

To minimize the contribution of VCO phase noise, the loop bandwidth of the PLL should be made large. This is in clear contradiction to the situation with reference noise discussed in Sec. 6.7.1. Here, the loop bandwidth had to be made as large to minimize the contribution of reference phase noise. To optimize overall noise at the PLL output, a compromise must be found for loop bandwidth.

## Spurs created by the phase detector

The frequency synthesizer shown in Fig. 3.1 generates an output signal whose frequency is exactly  $N$  times the reference frequency. In the ideal case, the VCO would operate at its center frequency. The output signal of the loop filter would then be exactly zero. Under these conditions, the output signal of the VCO would be a “pure” square wave (or sine wave)—in other words, a signal without any phase jitter. The spectrum of this signal would consist of a line at  $f=f_0$  and—as is normal for a rectangular signal—a number of harmonics.

Under normal PLL operation, however, the phase detector delivers correction signals as soon as the phase of the VCO output signal deviates from the ideal phase. If an EXOR gate is used as phase detector, its output signal is a square-wave signal whose frequency is twice the reference frequency, as shown in Fig. 2.6. Because the gain of every loop filter is nonzero at that frequency, a *ripple signal* appears at the input of the VCO. The ripple signal modulates the output signal

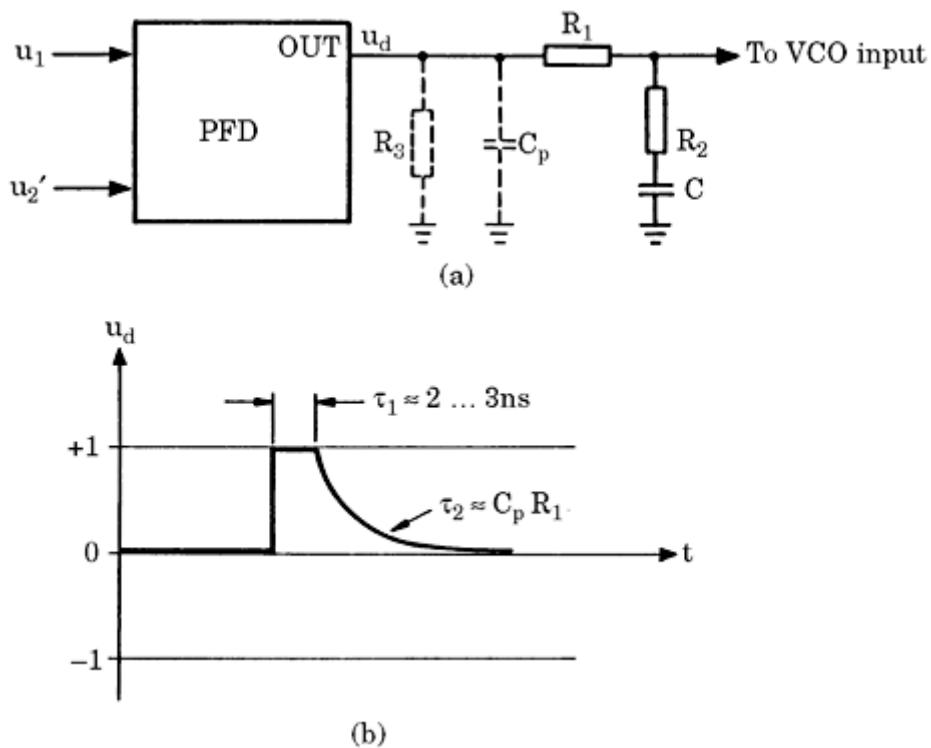
**Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.**

of the VCO. As we will see, this can lead to spurious sidebands in the VCO output signal, which affects its spectral purity, of course. The effects of the ripple are different for the various types of phase detectors and must therefore be treated separately.

Let us look first at what can happen when the EXOR phase detector is used. In a PLL frequency synthesizer (Fig. 3.1) with divider ratio  $N = 1$ , the instantaneous phase of the VCO output signal would be modulated by a signal whose frequency is twice that frequency. Obviously, this cannot have an impact on the VCO frequency. When  $N = 2$ , nothing happens again, because now the instantaneous phase of the VCO output signal is modulated by a signal whose frequency is identical with the VCO frequency. But adverse effects start as soon as  $N$  becomes 3 or larger. For general  $N$ , the frequency of the VCO output signal is  $N \cdot f_{\text{ref}}$ , and the ripple signal has a fundamental frequency of  $2f_{\text{ref}}$ . For  $N = 3$ , spurious sidebands appear at the VCO output whose frequencies are  $N \cdot f_{\text{ref}} \pm 2f_{\text{ref}}$ ,  $N \cdot f_{\text{ref}} \pm 4 \cdot f_{\text{ref}}$ , and so on. If the multiplier phase detector is used instead of the EXOR, spurs will be created at the same frequencies.

The situation is similar when the JK-flipflop is used as a phase detector. In the locked state, the JK-flipflop generates an output signal whose frequency is identical with the reference frequency (see Fig. 2.9). If the divider ratio of the frequency synthesizer is 1, the ripple on the VCO input signal only alters the duty cycle of the VCO output signal but does not generate spurious sidebands. Spurious signals appear, however, when  $N$  becomes 2 and greater. Then the spectrum of the output signal contains lines at frequencies  $N \cdot f_{\text{ref}} \pm f_{\text{ref}}$ ,  $N \cdot f_{\text{ref}} \pm 2f_{\text{ref}}$ , and so on.

When the PFD with voltage output is used as a phase detector, spurious side-bands at a subharmonic of the reference frequency can show up, which is very disturbing because there is a good chance for those spurs to lie inside the loop bandwidth  $B_L$ . These spurs are created by a phenomenon called backlash, which will be described in the following. When the PLL operates exactly on its center frequency, the output signal of the PFD is theoretically in the 0 state all the time. The reference signal  $u_1$  and the (scaled-down) output signal  $u_2'$  (refer to Fig. 6.21a) would then be exactly in phase. In reality, the frequency of the VCO output signal will slowly drift away, which causes a time lag between these two signals. When this time lag is 10 ps, for example, the PFD will theoretically generate a correction pulse whose duration is 10 ps as well. Because the logical circuits inside the PFD have nonzero propagation delays and rise times, the PFD is never capable of generating such short pulses. It will produce a correction pulse only when the delay between the  $u_1$  and  $u_2'$  signals has become greater than a time interval called *backlash*. For high-speed CMOS circuits, the backlash is typically in the range of 2 to 3 ns. The PFD never generates an output pulse shorter than the backlash interval. Fig. 6.21b shows the PFD output signal  $u_d$ , which has been generated just at the instant where the positive edge of the  $u_1$  signal led the positive edge of  $u_2'$  by an amount equal to the backlash, denoted as  $\tau_1$  here. The width of the correction pulse is nearly equal to  $\tau_1$  in this case. Unfortunately, each real device contains parasitic capacitances. In our example, parasitic



**Figure 6.21** The backlash effect of the PFD and the effect of parasitic capacitance at the phase detector output. (a) Schematic of the PFD including parasitic capacitor  $C_p$ . The loop filter is also shown. (b) Waveform of the  $u_d$  signal. The duration  $\tau_1$  of the output pulse cannot be less than the backlash interval.

capacitance  $C_p$  (Fig. 6.21a) will charge to the supply voltage by the correction pulse, so the decay of that pulse will slow down. Typically,  $C_p$  is in the order of 5 to 10 pF. The time constant of the decay is approximately  $\tau_2 = R_1 C_p$ . It can be made small by selecting a low value for resistor  $R_1$ , but  $R_1$  cannot be chosen arbitrarily low, because this would overload the PFD output. Typically,  $R_1$  must be higher than about 500  $\Omega$ . Thus,  $\tau_2$  will also be on the order of 5 ns or more. Because the correction pulses cannot be arbitrarily narrow, the pulse as depicted in Fig. 6.21b alters the instantaneous frequency of the VCO more than would normally be required. In the example of Fig. 6.21b, the frequency of the VCO is increased. Therefore, the time delay between the edges of the  $u_1$  and  $u_2'$  becomes shorter in succeeding cycles of the reference signal. After some cycles, the delay crosses zero and becomes negative, so that  $u_2'$  leads  $u_1$  now. The PFD will only produce a correction pulse (of negative polarity), however, when the time lag exceeds the backlash again. This leads to the unhappy situation that the PFD generates a positive correction pulse at some instant, stays in the zero state during a number of succeeding cycles (perhaps 10), produces a negative correction pulse then, and so forth. The frequency of the ripple signal is therefore a *subharmonic* of the reference signal, and if the frequency of this subharmonic is very low, the ripple is not attenuated by the loop filter, which is undesirable, of course. It is possible to calculate approximately the subharmonic frequency, using the formulas given in ref. 15. Usually, the

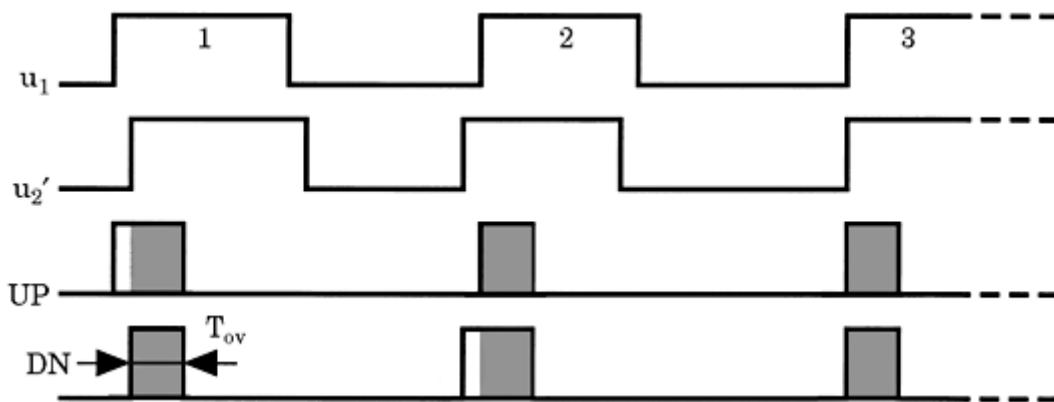
subharmonic frequency is about 1/20 the reference frequency.

We can eliminate the generation of subharmonic spurs by forcing the PFD to operate at a nonzero phase error all the time. As shown in Fig. 6.21, a high-value resistor  $R_3$  slowly discharges capacitor  $C$  toward ground. If the discharging current is higher than the charging current produced by the parasitic capacitance, the PFD is forced to generate positive correction pulses *in every cycle* of the reference signal. The ripple frequency is now identical with the reference frequency. Because the cutoff frequency of the loop filter is lower than the reference frequency in most cases, the ripple signal will be attenuated by the loop filter, which decreases the level of the spurious sidebands. When resistor  $R_3$  is used, the spurious sidebands occur at frequencies  $N \cdot f_{\text{ref}} \pm f_{\text{ref}}$ ,  $N \cdot f_{\text{ref}} \pm 2f_{\text{ref}}$ , and so on.

The backlash problem can be almost entirely eliminated when the PFD with current output is used (refer also to Sec. 2.4.4.2). Such a PFD is included, for example, in the integrated circuit type 74HCT9046A manufactured by Philips<sup>51</sup>. This IC contains two phase detectors, an EXOR and a PFD. The latter has a charge pump output. As explained in Sec. 2.4.4.2, the timing of both current sources is modified such that the outputs of the upper and lower current sources (cf. Fig. 2.16) overlap during an interval of approximately 15 ns—thus, both current sources are turned on for at least 15 ns in every reference cycle. The timing of the current outputs is demonstrated by Fig. 6.22 for three cases:

- The signal  $u_1$  leads the signal  $u_2'$
- The signal  $u_2'$  leads the signal  $u_1$
- Both positive edges of  $u_1$  and  $u_2'$  occur “at the same time”

These cases are labeled 1, 2, and 3, respectively, in the figure. In case 1, the leading edge of  $u_1$  unconditionally sets the UP flipflop. On arrival of the leading edge of  $u_2'$ , the DN flipflop is also set unconditionally. Both flipflops are set now. In a conventional PFD, both flipflops would be reset immediately by the AND gate, which drives the  $C_D$  (clear direct) inputs of the D flipflops, hence the DN flipflop would be set during an extremely short interval only. Delaying



**Figure 6.22** Pulse timing of the PFD in the 74HCT9046A circuit. The shaded areas characterize the overlapping interval ( $T_{\text{ov}}$ ), where  $T_{\text{ov}} \approx 15$  ns.

**Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.**

the reset action by about 15 ns makes sure, however, that both flipflops remain in their “high” state for 15 ns after the leading edge of  $u_2'$ . In case 2, where  $u_2'$  leads  $u_1$ , the leading edge of  $u_2'$  first sets the DN flipflop, and the UP flipflop is set later on the leading edge of  $u_1$ . Now a similar thing happens: delaying the reset of both flipflops by 15 ns lets both flipflops set for about 15 ns after the leading edge of  $u_1$ . Now assume that both positive transients of  $u_1$  and  $u_2'$  occur “at the same time.” Both flipflops are set immediately, and they remain set for about 15 ns.

With this kind of PFD, there is a guaranteed overlap of both current outputs in every reference cycle. This virtually inhibits the backlash effect as described for voltage output PFDs. To demonstrate this, assume that the positive edge of  $u_2'$  is delayed by 1 ns against the positive edge of  $u_1$ . In a conventional PFD, this time delay would fall below the backlash interval, hence the PFD would not perform any reaction. In the case of the 74HCT9046A, however, the upper current source in Fig. 2.16c turns on the positive edge of  $u_1$ . One ns later, the positive edge of  $u_2'$  turns on the lower current source as well. From now on, both current sources will be on for 15 ns. Consequently, the upper current source is on for 16 ns, and the lower for 15 ns. This results in a net positive charge supplied to the loop filter. If the  $u_2'$  signal would perform a positive transition 1 ns *before* the  $u_1$  signal, the lower current source would be turned on first. The upper current source would then turn on 1 ns later. From this instant on, both current sources would remain on for 15 ns, hence the lower would conduct current for 16 ns, the upper for 15 ns, which would result in a net negative charge delivered to the loop filter. This demonstrates that even for the smallest time delay between the leading edges of  $u_1$  and  $u_2'$ , the PFD always feeds some charge into the loop filter. The pulse widening due to parasitic capacitances (as shown in Fig. 6.21) does not play any role for this type of PFD because both current sources are turned on in every reference cycle, thus canceling the widening effect.

There is nevertheless one source of error observed with charge pump PFDs: the imbalance of the current sources<sup>53</sup> (cf. Fig. 2.16). When the source current does not exactly match the sink current and both current sources are ON during an identical interval of time, the charges do not compensate to zero but there is a nonzero charge delivered to the output of the PFD. When this happens, the PLL is forced to operate with nonzero phase error  $\pi_e$ . Consequently, the PFD must output a short output pulse in every reference cycle, and this leads to a spur at the reference frequency  $f_{\text{ref}}$ .

To conclude, we give a couple of equations enabling us to estimate the spurs to be expected. Because the situation depends very much on the type of phase detector used, the equations are different for different phase detectors. Because the amplitude of the first pair of sidebands is always much greater than the amplitude of the higher ones, it is sufficient for practical purposes to have an approximation for the first sideband. In analogy to noise signals, sideband suppression  $S_1$  has been defined as the quotient of signal power to the first sideband power:

$$S_1 = \frac{P_s}{P_{\text{sb1}}} \quad (6.34)$$

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

where  $P_s$  is the power of the synthesized signal and  $P_{\text{sb1}}$  is the power of the first spurious sideband. Mostly, a logarithmic quantity  $S_{1(\text{dB})}$  is specified:

$$S_{1(\text{dB})} = 10 \log S_1 \quad (6.35)$$

If the EXOR or the JK-flipflop is used, an approximation for  $S_{1(\text{dB})}$  is given by<sup>15</sup>

$$S_{1(\text{dB})} = 20 \log \frac{K_0 U_B}{2\pi^2 f_r} |F(2\pi f_r)| \quad (6.36)$$

Here  $f_r$  denotes ripple frequency.  $F(2\pi f_r)$  is the gain of the loop filter at the ripple frequency, and  $U_B$  is the supply voltage of the phase detector. In the case of the EXOR gate,  $f_r$  equals twice the reference frequency.

When the JK-flipflop is used, however,  $f_r$  is identical with the reference frequency.

For the PFD with voltage output, another approximation has been found:<sup>15</sup>

$$S_{1(\text{dB})} = 20 \log(K_0 U_B \tau) |F(2\pi f_r)| \quad (6.37)$$

Here, the ripple frequency  $f_r$  is usually a subharmonic of the reference frequency  $f_{\text{ref}}$ , as described earlier.  $\tau$  is the duration of PFD output pulses, as shown in Fig. 6.21. For  $\tau$ , we have approximately

$$\tau \approx \tau_1 + \tau_2 \quad (6.38)$$

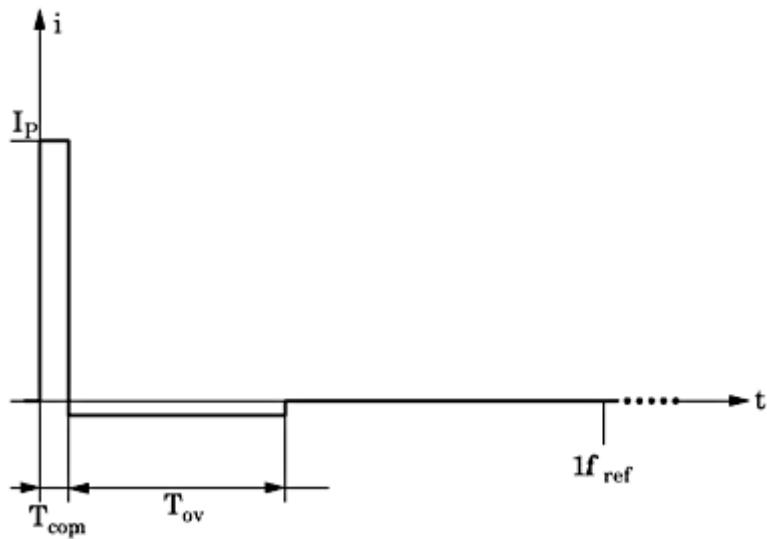
with  $\tau_2 = R_1 C_p$  [cf. Fig. 6.21].

The charge pump PFD is optimum with respect to spurs. If the current sources were perfectly matched, there would be theoretically no spurs at all. An imbalance of the current sources leads to a spur, however, at the reference frequency. The following analysis leads to an approximate formula for the power of this spur.

Assume that the imbalance of the current sources is  $\delta$ —that is, one of the current sources of the PFD (cf. Fig. 2.16) supplies current  $I_P$ , the other  $(1-\delta)I_P$ . When the overlapping interval (as defined in Fig. 6.22) is  $T_{\text{ov}}$ , a net charge of  $Q = \delta I_P T_{\text{ov}}$  flows into the load in every reference cycle. This charge must be compensated for by a finite phase error. When the sink current is the larger one,  $Q$  becomes negative, and the phase error must become positive so a positive charge will be supplied in the interval  $T_{\text{comp}}$ , as shown in Fig. 6.23. Because the area under the positive pulse must equal the area under the negative pulse, we have

$$I_P T_{\text{comp}} = \delta I_P T_{\text{ov}} \quad (6.39)$$

The current waveform shown in [Fig. 6.23](#) creates an AC signal  $u_{f,\text{ac}}$  at the output of the loop filter (cf. [Fig. 2.17b](#) for a first-order loop filter), and this AC signal will modulate the frequency of the VCO, thus generating spurs at multiples of the reference frequency  $f_{\text{ref}}$ . To get an estimate of these spurs, we restrict



**Figure 6.23** Output current versus time for a charge pump PFD.

ourselves on calculating the fundamental only—in other words, the spur at an offset  $f_{\text{ref}}$  from the carrier frequency  $f_0$ . First, we want to compute the fundamental of the current waveform in Fig. 6.23. When we assume that the time intervals  $T_{\text{ov}}$  and  $T_{\text{comp}}$  are much shorter than the reference period  $1/f_{\text{ref}}$ , the Fourier series yields the absolute value of the fundamental  $k_1$

$$k_1 \approx I_P f_{\text{ref}}^2 \pi T_{\text{ov}}^2 \delta \quad (6.40)$$

Next, we have to compute the fundamental of the AC signal at the output of the loop filter. Assuming that a first-order loop filter (Fig. 2.17b) is used, we have

$$u_{f,\text{ac},1} = F(f_{\text{ref}}) k_1 \sin(\omega_{\text{ref}} t) \quad (6.41)$$

with  $u_{f,\text{ac},1}$  = fundamental of AC signal and  $\omega_{\text{ref}} = 2\pi f_{\text{ref}}$  and  $F(f_{\text{ref}})$  = transfer function (transfer impedance) of the loop filter. In most cases, the reference frequency is much higher than  $1/\tau_2 = 1/R_2 C_1$ , thus  $F(f_{\text{ref}}) \approx R_2$ . We can now compute the modulated radian frequency  $\omega_2(t)$  generated by the VCO

$$\omega_2(t) = \omega_0 + K_0 R_2 k_1 \sin(\omega_{\text{ref}} t) \quad (6.42)$$

By integration over time we get the phase  $\varphi_2(t)$  of the VCO output signal

$$\varphi_2(t) = \omega_0 t - K_0 R_2 k_1 \frac{\cos(\omega_{\text{ref}} t)}{\omega_{\text{ref}}} \quad (6.43)$$

From  $\varphi_2(t)$ , we can calculate the VCO output signal  $u_2(t)$  by setting

$$u_2(t) = U_0 \sin(\varphi_2[t]) \quad (6.44)$$

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).

Copyright ©2004 The McGraw-Hill Companies. All rights reserved.

Any use is subject to the Terms of Use as given at the website.

with  $U_0$  = amplitude of the fundamental of the VCO output signal. After a number of trigonometric manipulations, we finally get

$$u_2(t) = U_0 \sin \left[ \omega_0 t - \frac{K_0 R_2 I_P f_{\text{ref}} T_{\text{ov}}^2 \delta}{4} (\cos(\omega_0 + \omega_{\text{ref}})t + \cos(\omega_0 - \omega_{\text{ref}})t) \right] \quad (6.45)$$

Apparently  $u_2(t)$  consists of a carrier at radian frequency  $\omega_0$ , plus two side-bands displaced by  $\pm f_{\text{ref}}$  from the carrier. We can therefore calculate the carrier to sideband ratio  $S_1$  at frequency offset  $f_{\text{ref}}$  from

$$S_1 = \frac{\text{carrier power}}{\text{sideband power}}$$

Using the relations

$$I_P = K_P 2\pi \quad \begin{matrix} \text{[cf.} \\ (2.27)] \end{matrix} \quad \text{Eq.}$$

$$\tau_2 = R_2 C_1 \quad \begin{matrix} \text{(cf.} \\ 2.17b) \end{matrix} \quad \text{Fig.}$$

$$\omega_n^2 = \frac{K_P K_o}{N C_1} \quad \begin{matrix} \text{[cf.} \\ (3.23)] \end{matrix} \quad \text{Eq.}$$

$$\zeta = \frac{\omega_n \tau_2}{2} \quad \begin{matrix} \text{[cf.} \\ (3.23)] \end{matrix} \quad \text{Eq.}$$

we finally get

$$S_1 = \frac{1}{2\pi^2 \omega_n^2 \zeta^2 f_{\text{ref}}^2 T_{\text{ov}}^4 \delta^2 N^2} \quad (6.46)$$

We recognize that the carrier-to-spur ratio decreases with the square of the current imbalance  $\delta$ .

We conclude this chapter with a note regarding suppression of spurs by the loop filter. As we know, spurs mostly occur at the reference frequency and multiples thereof. They are easily suppressed when the loop bandwidth ( $f_{3\text{dB}}$ ) is chosen to be less than the reference frequency. Second-order PLLs, as they were analyzed in [Chap. 3](#), show a relatively flat gain rolloff at higher frequencies, because the phase-transfer function  $H(s)$  has two poles and one zero. Consequently, the gain rolls off with only  $-20$  dB/decade at higher frequencies. The spurs are therefore only partially suppressed. A better spur suppression is obtained when higher-order

loop filters are used. When applying a fourth-order loop filter, for example, the gain rolls off at  $-80$  dB/decade. Higher-order loops will be discussed in greater detail in [Chap. 9](#).

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

## Mixed-Signal PLL Applications Part 2: Fractional-N Frequency Synthesizers

### Realization of Fractional Divider Ratios

In the design example of [Sec. 6.5](#), we realized a PLL frequency synthesizer capable of creating output frequencies that are an integer multiple of the reference frequency (10 kHz). The synthesizer had a lock-in time of about 2 ms. This is not extremely fast. Actually, there is an empirical relation between lock-in time  $T_L$  and reference frequency  $f_{\text{ref}}$ , which says that the lock-in time  $T_L$  equals a number of reference periods, typically 10 to 20 reference periods (a reference period has the duration  $1/f_{\text{ref}}$ ). [10.48](#) Where does that range stem from?

In most practical PLL designs, the down-scaled center frequency  $\omega_0'$  is much larger than the natural frequency  $\omega_n$  of the PLL. Typically,  $\omega_0'$  is about 20 times the natural frequency. Note that in a PLL frequency synthesizer the down-scaled center frequency  $\omega_0'$  is identical with  $2\pi f_{\text{ref}}$ —in other words, the reference period equals one period of down-scaled center frequency. Also remember that the lock-in time is approximately equal to one period of natural frequency—that is

$$T_L \approx \frac{2\pi}{\omega_n}$$

[cf. [Eq. \(3.62\)](#)]. Because the down-scaled center frequency is larger by a factor of 20 than the natural frequency, the lock-in time corresponds to roughly 20 reference periods.

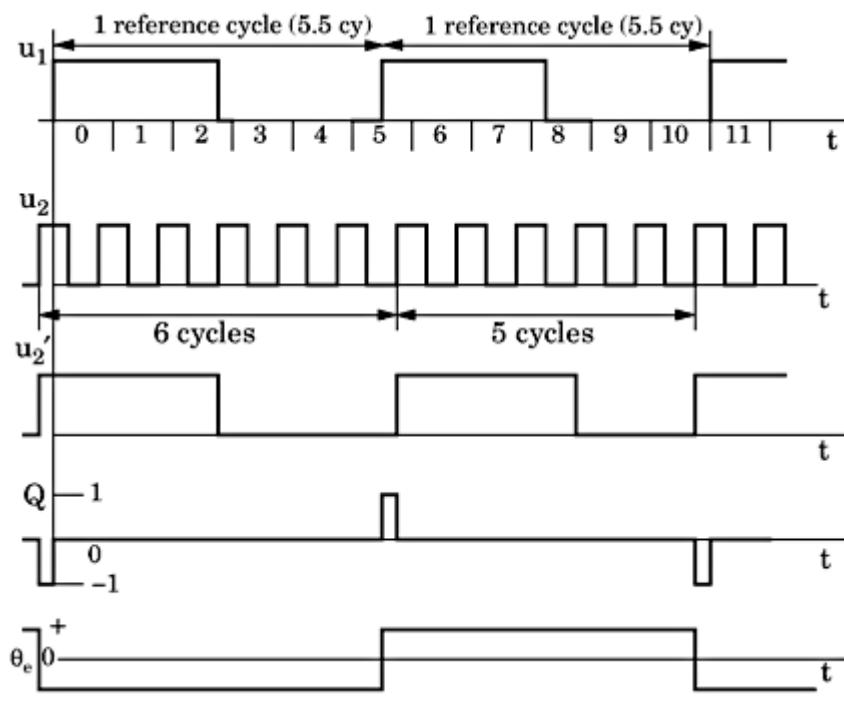
When the channel spacing of a frequency synthesizer must be narrow (for example, on the order of 1 kHz), we are forced—at least in conventional synthesizer circuits—to choose a low reference frequency. As we have seen, this results in a slow lock-in time. In a conventional synthesizer (that is, in the type of synthesizer we considered hitherto) the output frequency has always been an integer multiple of the reference frequency—for example,  $10 \cdot f_{\text{ref}}$ ,  $11 \cdot f_{\text{ref}}$ ,  $12 \cdot f_{\text{ref}}$ , and so on. Assuming  $f_{\text{ref}} = 10$  kHz, we could create frequencies of 100 kHz, 110 kHz,

120 kHz, and up. Let's now think about a divide-by-N counter that is not only able to scale down by 10, 11, and 12, but by 10.0, 10.1, 10.2, and so on. If such a down scaler were realizable, we could create output frequencies of  $10 \cdot f_{\text{ref}}$ ,  $10.1 \cdot f_{\text{ref}}$ ,  $10.2 \cdot f_{\text{ref}}$ , and so forth. Still using  $f_{\text{ref}} = 10$  kHz, we now would be able to create the frequencies 100 kHz, 101 kHz, 102 kHz, and so on. To obtain a channel spacing of 10 kHz, we now could choose  $f_{\text{ref}} = 100$  kHz, which is the tenfold of the "old" reference frequency! Doing so, the natural frequency could also be increased by a factor of 10, and the loop would be ten times as fast as the "old" one.

This is one of the ideas behind the so-called fractional-N frequency synthesizer. Other reasons lead to the design of fractional-N synthesizers, too, of course: there could be a demand to produce signals whose frequency can be any multiple of a precise frequency standard—for example, 123.45 kHz, 146.73 kHz, and so on. Signal generators are an example for those applications.

Since a down scaler is always a digital circuit, it certainly can't divide by fractional numbers like 10.1 or 10.2, only by integer numbers like 10 or 11. Dividing by 10.5, for example, becomes possible, however, if the divide-by-N counter is made to scale down alternately by 10 and by 11. On average, this counter effectively divides the input frequency by 10.5. Dividing by 10.1 also becomes realizable, if the counter divides by 11 in one cycle within a sequence of 10 cycles, and by 10 in the remaining 9 cycles. Switching the divider ratio of a down scaler is simple to realize, but this technique inevitably leads to phase jitter, as will be demonstrated by the following example.

Assume that we want to implement a divider ratio of 5.5. This is accomplished by letting the down scaler divide by 5, then by 6, then by 5, and so on, as shown in Fig. 7.1. The topmost trace shows the reference input  $u_1$  of the



**Figure 7.1** A realization of the fractional division ratio by switching the preset count of

the down scaler.

---

**Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.**

frequency synthesizer; each reference cycle has a duration corresponding to 5.5 cycles of the output signal  $u_2$ , which is shown in the second trace. The third trace represents the down-scaled output signal  $u_2'$ . In the first reference cycle shown, the down scaler divides by 6; hence, 6 cycles of  $u_2$  correspond to one reference cycle. In the second reference cycle, the down scaler divides by 5; hence, 5 cycles of  $u_2$  equal one reference cycle. We easily recognize that in the first reference cycle the  $u_2'$  signal leads the  $u_1$  signal by a quarter of an output cycle. In the second reference cycle, however, the  $u_2'$  signal lags  $u_1$  by a quarter of an output cycle and so forth. The fourth trace shows the state of the output signal of the phase-frequency detector ( $Q$ ). In the first reference cycle,  $Q$  becomes  $-1$  during a quarter of an output cycle—that is, the phase error becomes negative. In the second reference cycle,  $Q$  is set to  $+1$  during a quarter of an output cycle, and the phase error becomes positive. Consequently, the phase error  $\theta_e$  oscillates between positive and negative values in consecutive reference cycles—that is, the PFD output signal contains an AC component whose frequency is half the reference frequency. This leads to a spur that is displaced by  $f_{\text{ref}}/2$  from the carrier frequency  $f_0$ . It is easily shown that for other fractional division ratios, the spurs will appear at other locations in the frequency spectrum. For a division ratio of 5.2, for example, a spur will be displaced by  $f_{\text{ref}}/5$  from the carrier. For a division ratio of 5.1, a spur will be displaced by  $f_{\text{ref}}/10$ , and so on.

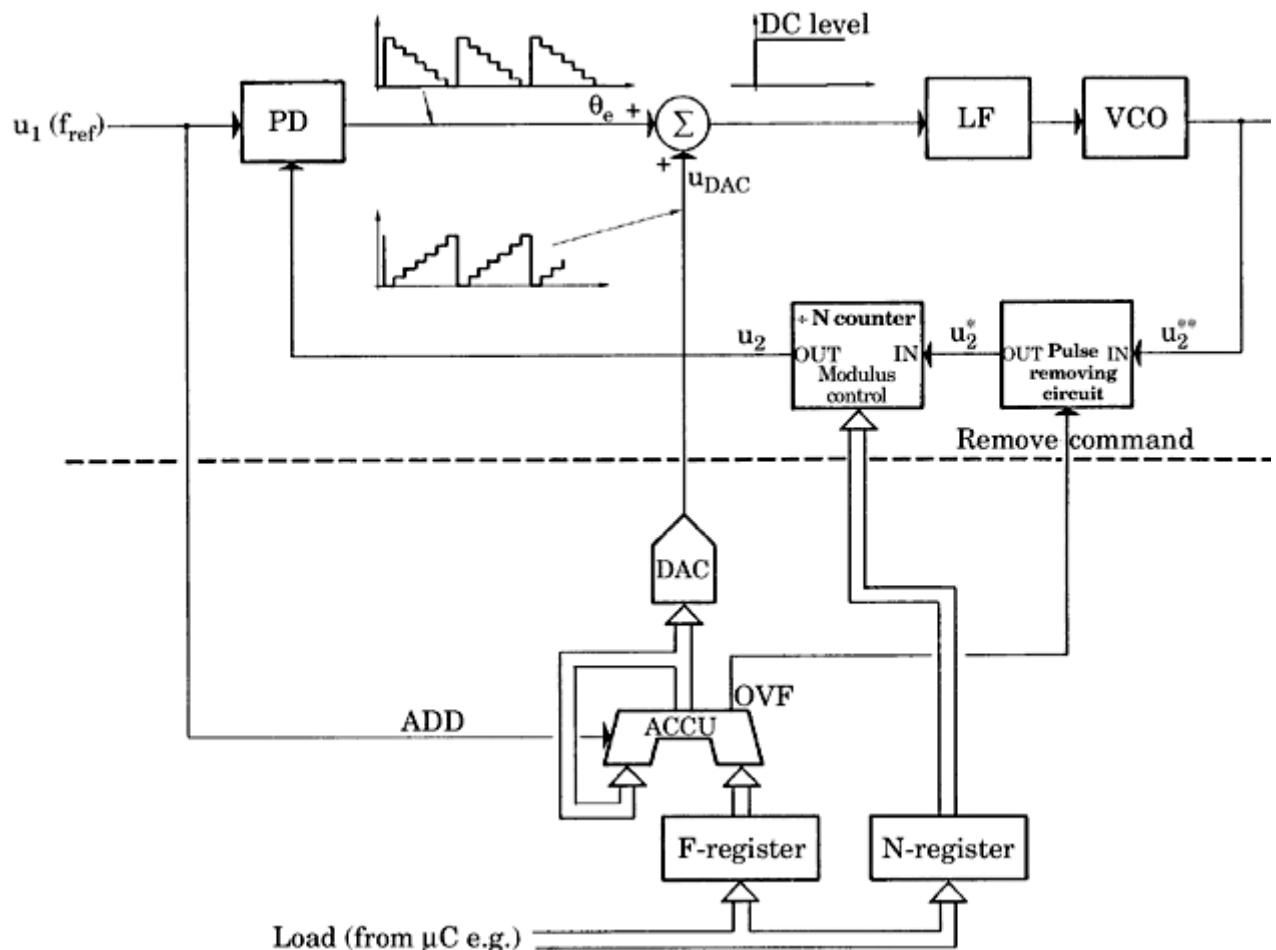
Of course, these spurs must be suppressed as much as possible. In the past, analog techniques have been used, as will be discussed in [Sec. 7.2](#). These methods are considered obsolete today; in the majority of modern synthesizer designs, digital methods dominate. They make use of the so-called Sigma-Delta modulator (also known as the *Delta-Sigma modulator*). Digital methods are preferred because they are much more easily implemented by integrated circuits than the former analog ones. Digital spur reduction methods will be described in [Sec. 7.3](#).

To obtain fractional divider ratios, we switched the preset count of down scalers from one value to another—for example, from  $N$  to  $N + 1$ . Another technique will lead to the same result: *cycle steering* or *pulse removal*. When making use of cycle steering, the down scaler of the frequency synthesizer always counts down by the same divider ratio. Instead of switching the preset count to  $N + 1$  in some reference cycles, one output pulse (cf. the  $u_2$  signal in [Fig. 7.1](#)) is removed from the counting input of the down scaler in those cycles. This is equivalent to down counting by  $N + 1$  in the corresponding reference cycle. In [Sec. 7.2](#), we will see an example of cycle steering.

## Analog Spur Reduction Techniques

[Figure 7.2](#) shows the block diagram of a fractional- $N$  frequency synthesizer. Its division ratio is the number  $N \cdot F$ , where  $N$  is the integer part and  $F$  is the fractional part. For example, if a division ratio of 26.47 is desired, then  $N = 26$  and  $F = 0.47$ . The upper portion of [Fig. 7.2](#), separated by a dashed line, shows an ordinary frequency-synthesizer system generating an output signal whose frequency is  $N$  times the reference frequency  $f_{\text{ref}}$ . The block labeled “pulse-removing circuit”

**Any use is subject to the Terms of Use as given at the website.**



**Figure 7.2** A fractional- $N$  frequency synthesizer block diagram.

and the summing block  $\Sigma$  should be disregarded for the moment. The integer and fractional parts ( $N$  and  $F$ , respectively) of the division ratio  $N \cdot F$  are stored in a buffer register as shown in the lower part of Fig. 7.2. The numbers  $N$  and  $F$  can be loaded via a serial or parallel data link from a microprocessor. The  $F$  register stores a fractional number;  $F$  can be stored in any code (binary, hexadecimal, BCD, and so on). If  $F$  is represented as a two-digit fractional BCD number, for example, the  $F$  register is an eight-bit register, where the individual states are assigned weights of 0.8, 0.4, 0.2, and 0.1 (tenths register), and 0.08, 0.04, 0.02, and 0.01 (hundredths register).

We now investigate how a frequency synthesizer can divide its output frequency  $f_{\text{out}}$  by fractional numbers. For example, let's see how the system generates an output frequency  $f_{\text{out}} = 5.3 \cdot f_{\text{ref}}$ . We can understand the operation of the fractional  $N$  loop by examining the waveforms shown in Fig. 7.3. Assume that the output frequency is already 5.3 times the reference frequency, and refer to the waveforms  $u_2^{**}$  (VCO output signal) and  $u_1$  (reference signal) in Fig. 7.3. The signal  $u_2^{**}$  shows 53 cycles during the time interval when the reference signal  $u_1$  is executing ten reference cycles. (In the following, the term *reference cycle* or *reference period* will be used to designate one full oscillation of the reference signal

$u_1$ .) Let the system start at the time  $t = 0$ .

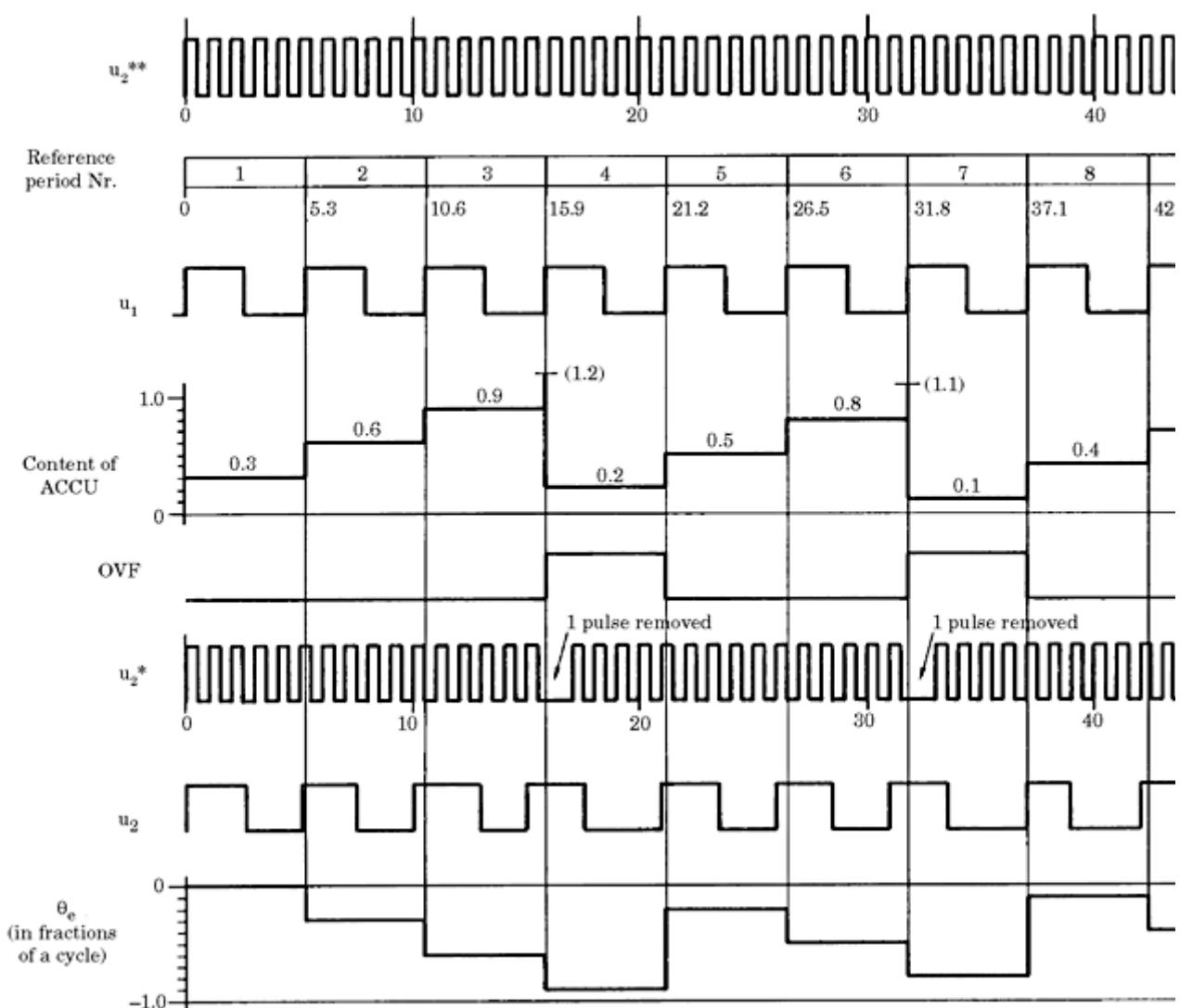


Figure 7.3 Waveforms explaining the operating principle of the fractional  $N$  loop.

During the time interval where  $u_1$  generates its first reference cycle, the  $\div N$  counter is required to divide the signal  $u_2^{**}$  by a factor of 5.3. This is impossible, of course, so the  $\div N$  counter will divide by 5 only. In the first reference period, 0.3 pulses are “missing.” This error (0.3) has to be memorized somewhere in the system; an accumulator (ACCU) is used for this purpose.

The ACCU uses the same code as the  $F$  register. If a two-digit fractional BCD format is used, the ACCU is capable of storing fractional BCD numbers within a range of 0.00 to 0.99. As seen in [Fig. 7.3](#), the ACCU adds the fractional number 0.F supplied by the  $F$  register to its original content whenever the ADD signal performs a positive transition, such as at the beginning of each reference period. If we assume that the initial content of the ACCU was zero at  $t = 0$ , the ACCU will accumulate an error of 0.3 cycles during the first reference period, indicating that the synthesizer has “missed” 0.3 pulses during the first reference period.

In the second reference period, the  $\div N$  counter is again required to divide by 5.3. Because this is not possible, it will continue to divide by 5 instead. Since it has already missed 0.3 cycles in the first reference period, the total error has now accumulated to 0.6 cycles. In the third reference period, the accumulated error is 0.9 cycles, and in the fourth reference period, it is 1.2 cycles. However, the ACCU cannot store numbers greater than 1; consequently, it overflows and generates an OVF signal ([Fig. 7.3](#)). The content of the ACCU is now 0.2, as seen in [Fig. 7.3](#). The OVF pulse generated by the ACCU causes the pulse removing circuit to become active, and the next pulse generated by the VCO is removed from the  $\div N$  counter. This pulse removal has the same effect as if the  $\div N$  counter divided by 6 instead of 5.

As [Fig. 7.3](#) shows, the ACCU overflows again in the seventh and tenth reference periods. Three pulses will therefore be removed from the  $\div N$  counter in a sequence of ten reference periods. Because the  $\div N$  counter divides by 5 forever,  $10 \cdot 5 + 3 = 53$  pulses are produced by the VCO during ten reference periods. This is exactly what was intended.

However, one problem has been overlooked. If the VCO oscillates at 5.3 times the reference frequency  $f_{\text{ref}}$ , it produces 5.3 cycles during one reference period. The PD in [Fig. 3.12](#) will consequently measure a phase error of  $-0.3$  cycle (or  $-0.3 \cdot 2\pi$  rad) after the first reference period in [Fig. 7.3](#). Thus, the phase error has *negative polarity* because the reference signal  $u_1$  lags the signal  $u_2$ .

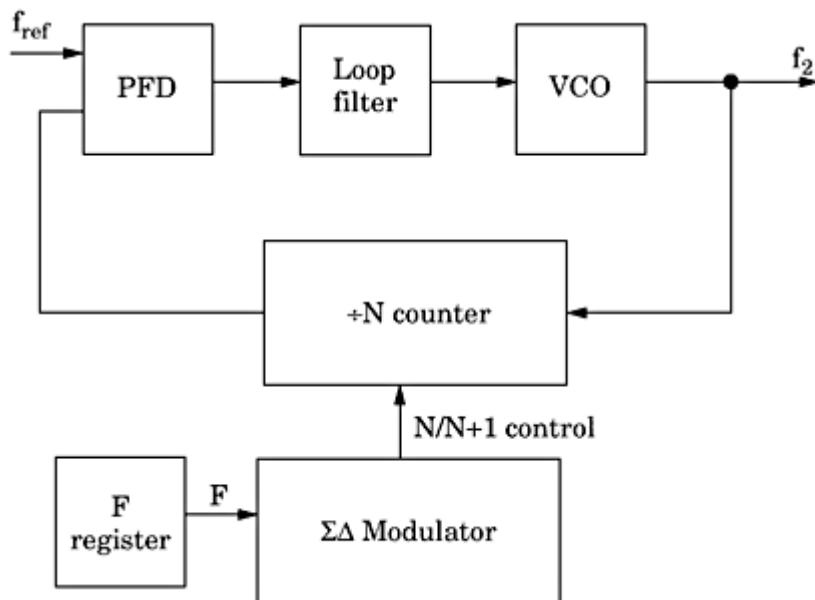
After the second reference cycle, the phase error has increased to  $-0.6$  cycles, and so on. The phase error  $\theta_e$  is plotted versus time in [Fig. 7.3](#); its waveform looks like a staircase. This phase error is applied to the input of the loop filter and will modulate the frequency of the VCO. Such a staircase-shaped modulation of the VCO frequency is not desired, however, because the pulse removing technique just discussed has already compensated for the phase error. There is an elegant way to avoid this undesired frequency modulation: The waveforms of [Fig. 7.3](#) show that the content of the ACCU has the same amplitude as the phase error  $\theta_e$  but with opposite polarity. The content of the ACCU is therefore converted to an analog signal by a DAC (digital-to-analog

converter); the output signal  $u_{\text{DAC}}$  is added to the output signal of the phase detector. Since the two staircase signals cancel each other, the input signal to the loop filter is a DC level when the fractional  $N$  loop has reached a stable operating point.

In the examples considered hitherto (cf. Figs. 7.1 and 7.2), the phase error showed up as a repetitive pattern. In the first case, it repeated every second reference cycle, while in the second example it repeated every tenth reference cycle. In the first case, a spur at half the reference frequency is created, while in the second case the spur occurs at 1/10 of the reference frequency. For other fractional parts  $F$ , the spurs can be at even smaller fractions of the reference frequency. Digital spur reduction techniques offer means for “randomizing” the phase error pattern. As a first consequence, the randomized error pattern is no longer periodic. Moreover, the randomization can be made such that the frequency spectrum of the phase error pattern contains most of its power at higher frequencies and almost zero power at frequencies near 0. Consequently, most of that phase noise will be removed by the loop filter. Displacing the power of the phase error spectrum toward higher frequencies is also called “noise shaping.” This will be discussed in greater detail in Sec. 7.3.

## Digital Spur Reduction Techniques

In modern fractional- $N$  synthesizer designs, practically only digital spur reduction techniques are applied. To realize fractional divider ratios, a Sigma-Delta modulator (referred to henceforth as SDM) is applied. Figure 7.4 shows the block diagram of a fractional- $N$  frequency synthesizer. It generates an output frequency  $f_2$ , which is  $N \cdot F$  times the reference frequency  $f_{\text{ref}}$ .  $N$  is the integer



**Figure 7.4** A fractional- $N$  frequency synthesizer using a sigma-delta modulator.

**Any use is subject to the Terms of Use as given at the website.**

part of the divider ratio, while  $F$  is the fractional part. Fraction  $F$  is stored in a register and is in the range of 0 to 1. The SDM creates a binary output signal in every reference cycle. If a logical 1 output signal is assigned the numerical value 1, and the logical 0 a numerical value of 0, then the numerical output signal becomes  $F$  on average. When  $F$  is 0.37, for example, the SDM output is 1 in 37 of 100 reference cycles on average. The output signal of the SDM is used to switch the divider ratio  $N$  of the programmable  $\div N$  counter from  $N$  to  $N + 1$ . Usually, the  $\div N$  counter is either a dual-modulus or a four-modulus counter, as discussed in Secs. [6.3.2](#) and [6.3.3](#). In the example shown, the output of the SDM is binary. Other cases exist, however, where the SDM outputs other types of digital signals—for example, ternary, quaternary, or in the most general case,  $m$ -ary. When the SDM output signal is  $m$ -ary, the divider ratio is not only switched between  $N$  and  $N + 1$ , but between  $m$  different values—for example,  $N - 3, N - 2$  thru  $N + 2, N + 3, N + 4$ .

The theory of sigma-delta modulators is quite complex, so we will discuss it in Sec. [7.4.3](#) in greater detail. Because SDMs have become popular originally in A/D and D/A converters, we will review those applications first before entering into details on SDMs in frequency synthesizers.

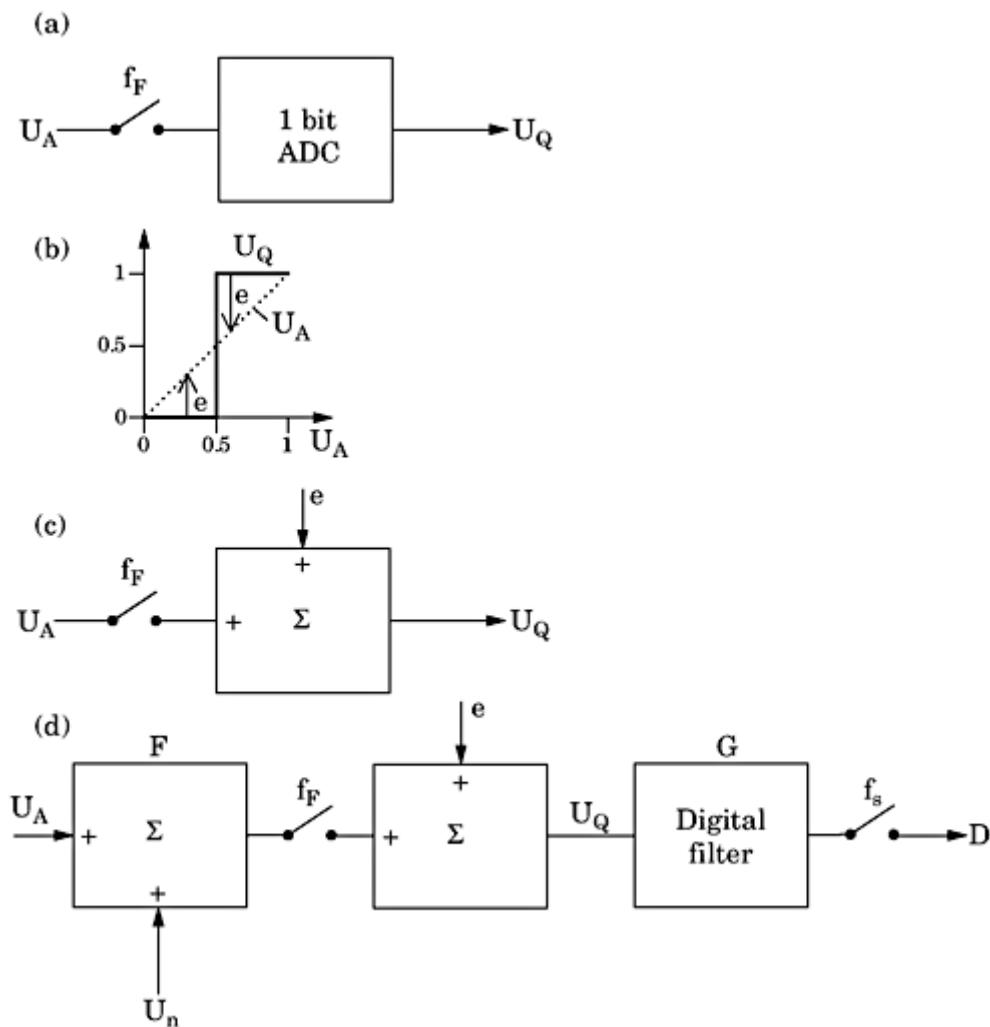
## Reviewing the $\Sigma\Delta$ Modulator

When designing a  $\Sigma\Delta$  modulator to be applied in a fractional- $N$  frequency synthesizer, we will encounter a number of particular problems such as loop stability, quantization noise, oversampling, and the like. To get familiar with these specific topics, we will discuss them in more detail using the example of the  $\Sigma\Delta$  ADC.

### The $\Sigma\Delta$ A/D converter

In the  $\Sigma\Delta$  ADC, oversampling is used to enhance the resolution (number of bits) of a low-resolution A/D converter. It then becomes possible to increase the resolution of an extremely simple one-bit ADC to 16 bits, for example, or even more. Several types of ADCs make use of oversampling; the  $\Sigma\Delta$  is just one possible configuration. We shall first explain the oversampling principle at an ADC configuration that is simpler than the  $\Sigma\Delta$  converter. The  $\Sigma\Delta$  ADC will be discussed thereafter.

Let's start with a one-bit ADC, as shown in [Fig. 7.5a](#). The analog input signal  $U_A$  is sampled at frequency  $f_F$  and applied to the ADC. The transfer characteristic of the one-bit ADC is plotted in [Fig. 7.5b](#). The analog signal  $U_A$  is supposed to be in the range of 0 to 1, which can be represented by a voltage range from 0 to 1 V or any other range. The ADC is nothing more than a quantizer. Its output signal  $U_Q$  is a quantized version of the input signal as sketched by the solid curve. The switching threshold of this quantizer is at  $U_A = 0.5$ . Input signals less than 0.5 are quantized as 0, and input signals larger than 0.5 as 1. The quantization step—the difference between the two logical levels of the quantizer—is  $Q = 1$  in this example.



**Figure 7.5** An example of a simple oversampling ADC. (a) A block diagram of a one-bit ADC. (b) The transfer characteristic of the one-bit ADC. (c) A mathematical model of the one-bit ADC. (d) A one-bit ADC with preprocessing ( $F$ ) and postprocessing ( $G$ ) function blocks.

The quantized output can now be represented as the sum

$$U_Q = U_A + e \quad (7.1)$$

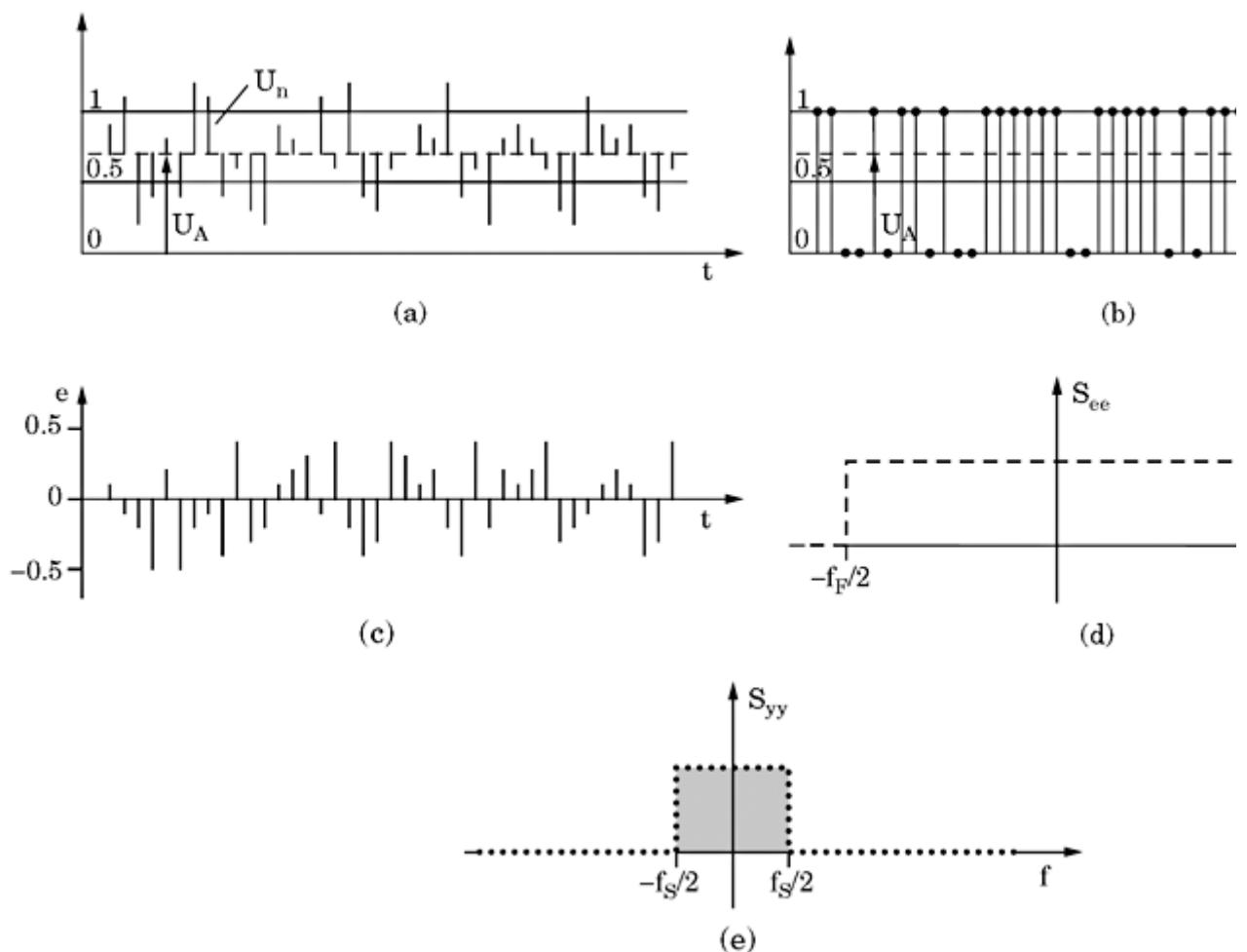
where  $e$  is called *quantization error*. This equation is shown graphically in Fig. 7.5b.  $U_A$  is the dotted curve, and  $e$  is represented by an arrow. As is easily seen, the quantization error is in the range

$$-Q/2 \leq e \leq Q/2 \quad (7.2)$$

thus,  $e$  is in the range  $-0.5$  to  $0.5$  in our example. When the input signal  $U_A$  varies with time, the quantization error forms an error sequence  $e(nT)$ , where  $T$  is the sampling interval ( $T = 1/f_F$ ) and  $n$  is a positive integer,  $n = 0, 1, 2$ , and so on. The error sequence plays an important role in oversampling ADCs. We will see that processing the error sequence is the key to increasing the number

of bits of the ADC. Using Eq. (7.1), we can now build a mathematical model of the one-bit ADC (see Fig. 7.5c). The one-bit converter is replaced here by a summing block representing Eq. (7.1). Assume for the moment that the input signal is constant,—for example,  $U_A = 0.7$ . What will the error sequence be in this case? The answer is trivial: Each quantized output sample has the value 1, and each quantization error becomes  $1 - 0.7 = 0.3$ . Processing the error doesn't bring any benefit, unless we manipulate the signal applied to the quantizer in such a way that the error sequence becomes non trivial, or, in other words, becomes a randomized sequence. Obviously, we must add at least two function blocks to the ADC—thus, a preprocessing block  $F$  and a postprocessing block  $G$ , as shown in Fig. 7.5d. This is the simplest oversampling ADC configuration, but be aware that it's not a  $\Sigma\Delta$  ADC. How will these two blocks look?

Block  $F$  is simply an adder where a uniformly distributed random signal  $U_n$  is added to the input  $U_A$ , where  $U_n$  is in the range  $-Q/2$  to  $Q/2$ . The mean of  $U_n$  must be zero. Fig. 7.6 shows the relevant signals of this A/D converter. In Fig. 7.6a, the input signal  $U_A$  and the (sampled) noise signal  $U_n$  are plotted. In the example,  $U_A$  is chosen as 0.7. In Fig. 7.6b, the quantized signal  $U_Q$  is plotted. The samples



**Figure 7.6** Waveforms of relevant signals in an oversampling ADC and their power density spectra

analog input signal  $U_A$  and the superimposed noise signal  $U_n$ . (b) The quantized output signal  $U_Q$  of ADC. (c) Error sequence  $e(nT)$ . (d) Power spectral density  $S_{ee}$  of the error sequence. (e) Power spectral density  $S_{yy}$  of the lowpass filtered error sequence.

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).

Copyright ©2004 The McGraw-Hill Companies. All rights reserved.

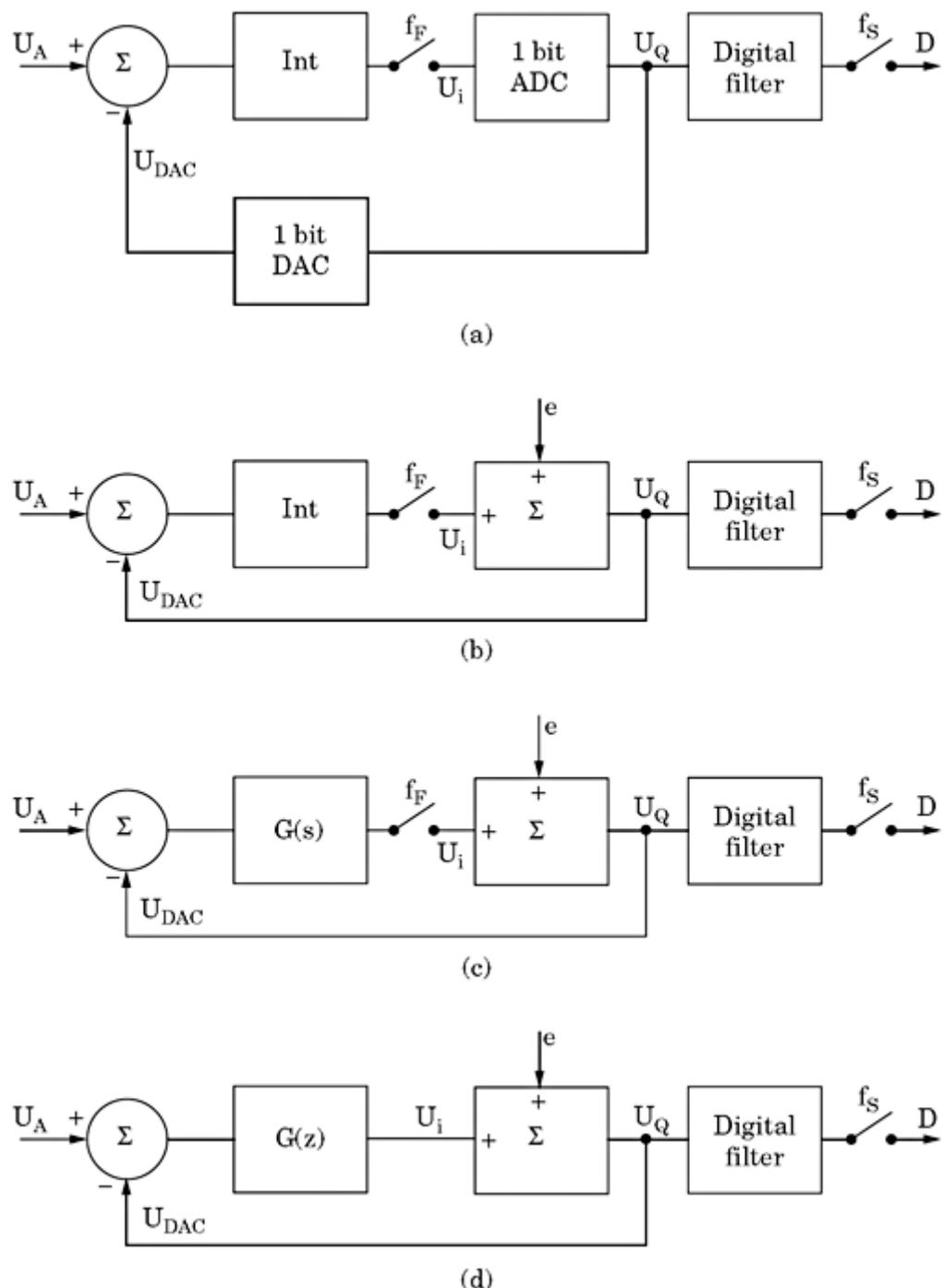
Any use is subject to the Terms of Use as given at the website.

are marked by dots. When observing  $U_Q$  during an extended period of time, note that  $U_Q$  is 1 in 70 percent of all samples, and 0 in the remaining 30 percent. The quantization error sequence  $e$  is plotted in Fig. 7.6c.  $e(nT)$  is now a random sequence with mean 0. To get rid of the ripple in the output signal of the quantizer, a digital lowpass filter is used as the postprocessing block  $G$  (Fig. 7.5d). This filter is used to decimate the output signal of the quantizer—thus, the output of the filter is read out at a lower sampling rate  $f_S$ . To avoid aliasing, the filter must remove all frequencies above  $f_S/2$ —in other words, the cutoff frequency filter must be  $f_S/2$  or less. The ratio  $f_F/f_S$  is called the *oversampling ratio* (OSR). What is the impact of that lowpass filter onto the quantized signal? To answer this question we must consider the power spectral density (PDS) of the error sequence  $e(nT)$ . Because  $e$  is purely random, the PDS of the error sequence is flat in the range  $-f_F/2$  to  $f_F/2$ , as shown in Fig. 7.6d. The PSD of the error sequence is denoted as  $S_{ee}$ . Due to lowpass filtering, all spectral components of  $S_{ee}$  above  $f_S/2$  (and below  $-f_S/2$ ) are removed. The PSD of the filtered error sequence is shown in Fig. 7.6e and is denoted as  $S_{yy}$ . Total quantization error power is reduced by a factor OSR. Consequently, the amplitude (peak value) of the lowpass filtered error sequence is reduced by the factor  $\sqrt{\text{OSR}}$ . Because we express the resolution of an ADC in numbers of bits, the so-called *bit gain*  $G$  is given by

$$G = \text{ld} \sqrt{(\text{OSR})} = \frac{1}{2} \text{ld}(\text{OSR}) \quad (7.3)$$

where  $\text{ld}$  signifies logarithm to the base 2. To make a numerical example, the bit gain  $G$  becomes 3 for an oversampling ratio of 64—thus, the resulting ADC now has a resolution of four bits. To obtain a bit gain of 15, however, the over-sampling ratio must be chosen as  $\text{OSR} = 2^{30} = 1,073,741,824$ , which is quite a large number! Obviously, there must be more efficient ways to increase ADC resolution. Let's therefore proceed to more powerful oversampling configurations: the  $\Sigma\Delta$  A/D converter.

Figure 7.7a shows the block diagram of the simplest configuration—a first-order  $\Sigma\Delta$  ADC. A one-bit ADC is preceded by an integrator (labeled “INT”). The quantized output signal  $U_Q$  which is a logical signal is converted back into analog format, and that analog signal is then fed back to a summing block on the left side of the schematic. It is assumed that all signals are in the same range as in the previously discussed A/D converter (Fig. 7.5d). In order to get a finite integrator output signal  $U_i$ , the difference  $U_A - U_{\text{DAC}}$  must be zero on average. Given  $U_A$ , the mean of  $U_{\text{DAC}}$  must equal  $U_A$ . When  $U_A$  is a DC level with value 0.5, for example,  $U_{\text{DAC}}$  will be a sequence 101010 ... The digital filter processing the output signal of the one-bit ADC is again a lowpass filter, whose corner frequency is  $f_F/(2 \text{ OSR})$  or less, where the oversampling ratio OSR is usually much greater than 1. To a first approximation, the output  $D$  of the digital filter is the average of the quantized signal  $U_Q$  (0.5 in this example) with a small superimposed ripple signal.



**Figure 7.7** Principle of operation of the  $\Sigma\Delta$  ADC. (a) A block diagram of a  $\Sigma\Delta$  ADC. (b) The same block diagram, but with the one-bit ADC replaced by its mathematical model. (c) Integrator replaced by its transfer function  $G(s)$  with  $s =$  Laplace operator. (d) A discretized version of the block diagram in (c), where the time-continuous model of the integrator has been replaced by a time-discrete model  $G(z)$  with  $z = z$  operator.

Next, we shall derive a mathematical model of this A/D converter (see Fig. 7.7b). As in the

previous example, the one-bit ADC is replaced by a summing block. The quantized output signal  $U_Q$  is considered to be the sum of input signal  $U_i$  and error sequence  $e$ . Because the DAC in the feedback path has a gain of 1, the output  $U_Q$  can now be connected directly with the inverting input of

the summing block on the left of the block diagram—thus, mathematically we have  $U_{\text{DAC}} = U_Q$ . Next, the integrator is replaced by its (time-continuous) transfer function  $G(s)$ , which is given by

$$G(s) = \frac{1}{sT_i} \quad (7.4)$$

with  $T_i$  = integrator time constant (see Fig. 7.7c). We realize now that the integrator is an analog function block, working in time-continuous mode, while the quantizer and the digital filter are digital blocks operating in discrete-time mode. The mathematical treatment becomes much simpler when the analog integrator is replaced by a digital one, as shown in Fig. 7.7d. Its transfer function then becomes

$$G(z) = \frac{T}{T_i} \frac{1}{1 - z^{-1}} \quad (7.5)$$

where  $T$  is the sampling interval,  $T = 1/f_F$ . ( $z$  is the so-called  $z$  operator, and  $G(z)$  is the  $z$  transfer function of the integrator; note that App. C is an introduction to  $z$  transform and digital filters.) To simplify things further, we set  $T/T_i = 1$  and get

$$G(z) = \frac{1}{1 - z^{-1}} \quad (7.6)$$

From the model in Fig. 7.7d, two transfer functions can now be derived that will be frequently used in the following. The first of these is the *signal transfer function* SFT( $z$ ) and is defined by

$$\text{STF}(z) = \frac{U_Q(z)}{U_A(z)}$$

It relates the quantizer output  $U_Q$  output to ADC input  $U_A$  assuming that the error  $e$  is zero. The second transfer function is called *noise transfer function* NTF( $z$ ) and is defined by

$$\text{NTF}(z) = \frac{U_Q(z)}{E(z)}$$

It relates the quantizer output  $U_Q$  to the error sequence assuming that input signal  $U_A$  is zero. The noise transfer function enables us to compute the bit gain of this type of ADC (as a function of OSR); this will be demonstrated in the following. Using the model in Fig. 7.7d, these transfer functions become

$$\text{STF}(z) = z^{-1} \quad (7.7)$$

$$\text{NTF}(z) = 1 - z^{-1} \quad (7.8)$$

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

The signal transfer function is just a delay by one sample. When the input signal is a DC level, the same DC level will show up at the quantizer output delayed by one sampling interval ( $T = 1/f_F$ ). This means practically that the input signal is transferred to the output with gain 1. The noise transfer function is simply the transfer function of a digital differentiator (difference builder)—that is, the output signal at discrete time  $t = nT$  is simply the difference  $e(nT) - e[(n-1)T]$ . When calculating the transfer functions as functions of frequency  $f$ , we must simply substitute  $z$  with  $e^{j2\pi fT}$  and produce some manipulations

$$|\text{SFT}(f)| = 1 \quad (7.9a)$$

$$|\text{NTF}(f)| = 2 \sin\left(\frac{\pi f}{f_F}\right) \quad (7.9b)$$

We are now able to compute the power spectral density of the quantization error at the output of the quantizer and digital filter. From this, we can further compute the bit gain of this type of A/D converter. In the ideal case, the error sequence  $e(nT)$  should be a random noise signal (as shown, for example, in Fig. 7.6c). As we will learn very soon, this is unfortunately not true for this type of  $\Sigma\Delta$  ADC. Later in this section, however, we will introduce some modifications that will “randomize” the error function so it becomes more or less “white.” Therefore, let’s assume for the moment that  $e(nT)$  is a random sequence; hence, its power density spectrum becomes flat in the frequency range  $-f_F/2$  to  $f_F/2$ . The power spectrum of the quantization error ( $S_{ee}$ ) is depicted in the uppermost trace in Fig. 7.8. To get the power density spectrum of the quantization error at the output of the lowpass filter, we must multiply  $S_{ee}$  by the square of  $|\text{NTF}(f)|$  and by the square of the transfer function of the lowpass filter, which is denoted here by  $H_{lp}(f)$ . Hence, we get

$$S_{yy}(f) = S_{ee}(f) |\text{NTF}(f)|^2 |H_{lp}(f)|^2 \quad (7.10)$$

The second trace in Fig. 7.8 shows the squared absolute value of  $\text{NTF}(f)$ , and the third trace illustrates the squared absolute value of  $H_{lp}(f)$ .

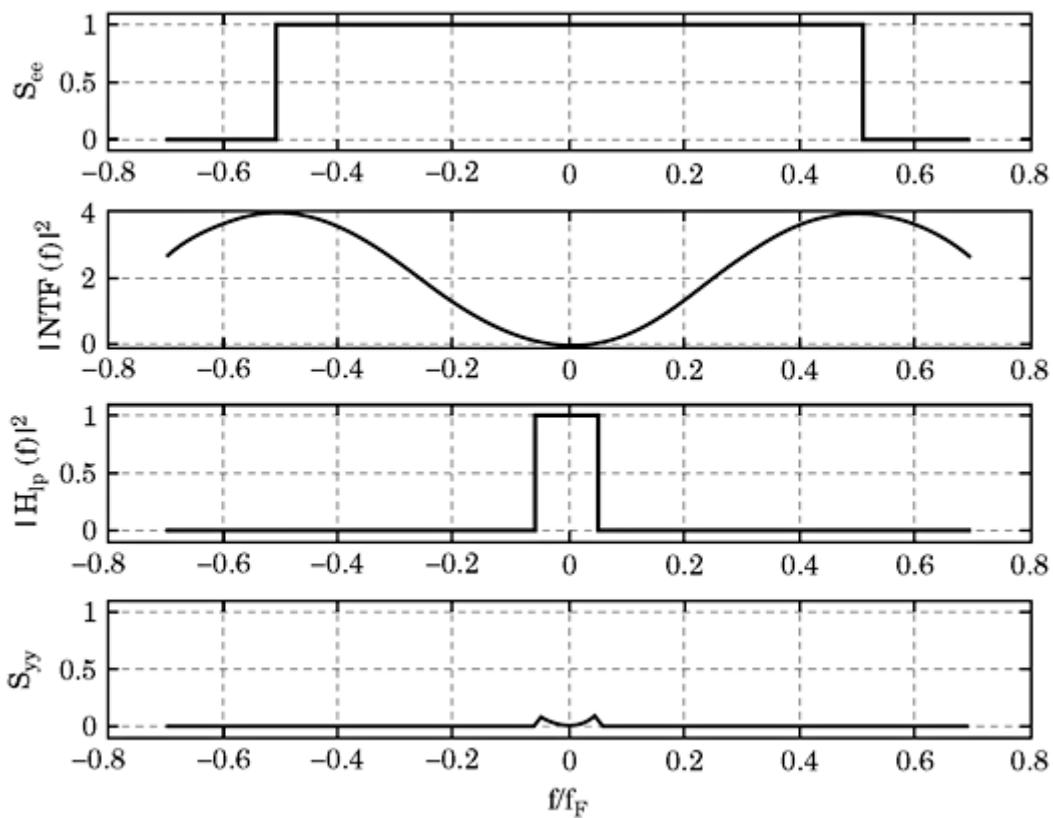
Performing the multiplications defined in Eq. (7.10) finally yields the attenuated power density spectrum  $S_{yy}$  at the output of the digital filter, shown in the last trace of Fig. 7.8. This result must be compared now with the result obtained for the simple ADC considered earlier (cf. Fig. 7.6e). Here, the total noise (that is, the area under the  $S_{ee}$  spectrum) was attenuated by the over-sampling ratio OSR. In the case of the  $\Sigma\Delta$  ADC, the attenuation is much larger because the gain of the differentiator [ $\text{NTF}(f)$ ] is near zero at low frequencies. The bit gain  $G$  of the first-order  $\Sigma\Delta$  ADC can be shown to be [55](#)

$$G = 1.5\text{ld(OSR)} + 0.5\text{ld}(3) - \text{ld}(\pi) \quad (7.11)$$

To get a bit gain of 15, for example, we would have to choose  $\text{OSR} = 1523$ , which compares favorably with the result obtained for the previously discussed

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.



**Figure 7.8** Frequency responses and power spectral densities related to first-order  $\Sigma\Delta$  ADC (cf. [Fig. 7.7](#))

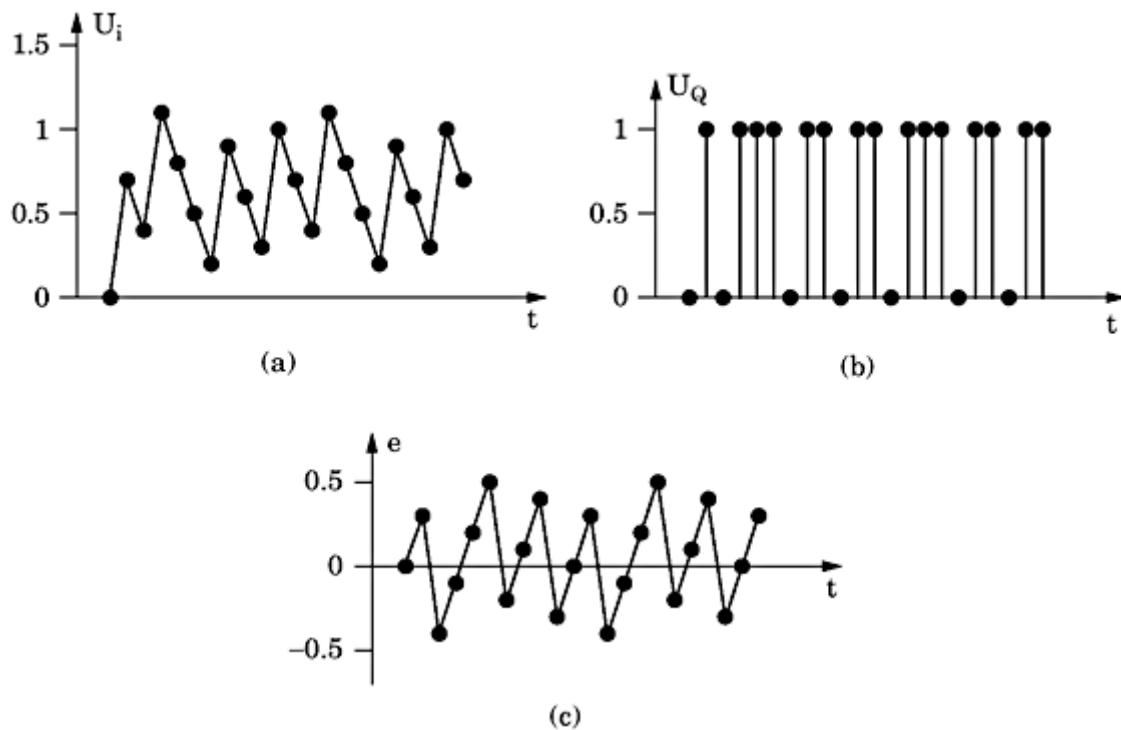
ADC ( $\text{OSR} = 1,073,741,824$ ). It should be noted that the error sequence passes through two filters in succession in this type of  $\Sigma\Delta$  ADC. The first filter is the differentiator NTF( $f$ ), which is essentially a highpass filter and suppresses the frequencies near zero. Because most of the noise power is now at higher frequencies, the action of the differentiator is also referred to as *noise shaping*. The second filter is the digital lowpass filter that suppresses the frequencies above its cutoff frequency.

As already mentioned, the error sequence of the  $\Sigma\Delta$  ADC (cf. [Fig. 7.7a](#)) is not a random sequence. We will have to find ways to “randomize” that sequence. Before going into detail, let’s have a look at the error sequence of the first-order  $\Sigma\Delta$  ADC. Assume that the input signal  $U_A$  is 0.7. [Figure 7.9a](#) shows the output signal  $U_i$  of the integrator ([Fig. 7.7a](#)). In [Fig. 7.9b](#), the quantized signal  $U_Q$  is plotted. Finally, the error sequence is drawn in [Fig. 7.9c](#). It is clearly seen that the sequence is periodic; it repeats itself every 10 samples—in other words, the spectrum of the error sequence has a fundamental at 1/10 of the sampling frequency  $f_F$  and harmonics at multiples thereof.

Whenever the input signal is a rational fraction (for example, 0.1, 0.2, 0.25, 0.333, 0.45, and so on) the error sequence shows a periodic pattern. There are two different methods to randomize the error sequence. One option is to use higher-order  $\Sigma\Delta$  converters—that is, converters having more than one integrator in the forward path (cf. [Fig. 7.10](#)).

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.



**Figure 7.9** Signals of the  $\Sigma\Delta$  ADC. (a) Signal  $U_i$  at the output of the integrator. (b) Output  $U_Q$  of the quantizer. (c) The error sequence  $e(nT)$ .

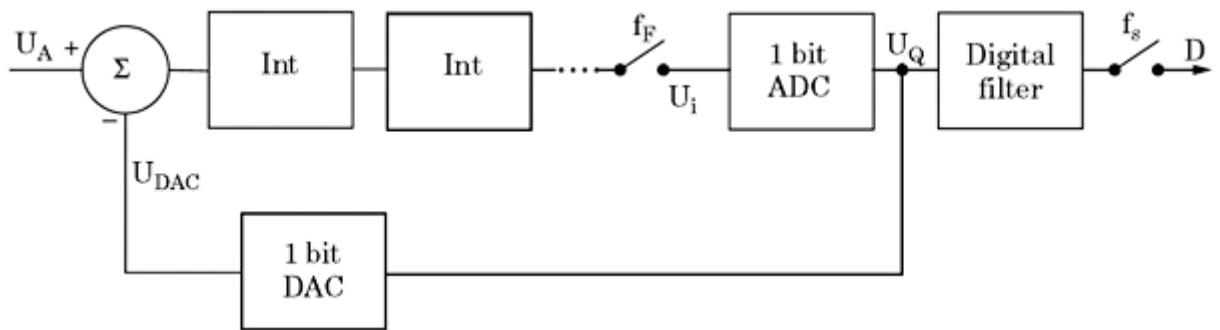
The transfer function  $G(z)$  of the integrator has been given in [Eq. \(7.6\)](#). When  $n$  integrators are cascaded, the transfer function of the series connection of integrators becomes

$$G(z) = \left[ \frac{1}{1 - z^{-1}} \right]^n \quad (7.12)$$

$n$  is also called the order of the system. For this type of  $\Sigma\Delta$  ADC, the signal and noise transfer functions are as follows:

$$\text{STF}(z) = z^{-n} \quad (7.13)$$

$$\text{NTF}(z) = (1 - z^{-1})^n \quad (7.14)$$



**Figure 7.10** A block diagram of higher-order  $\Sigma\Delta$  ADC.

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
 Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
 Any use is subject to the Terms of Use as given at the website.

The analog input signal  $U_A$  is simply delayed by  $n$  sampling intervals, and the error sequence is passed through  $n$  differentiators in cascade. The frequency response NTF( $f$ ) is therefore given by

$$|\text{NTF}(f)| = 2^n \sin\left(\frac{\pi f}{f_F}\right)^n \quad (7.15)$$

When comparing the frequency response of the  $n$ th-order  $\Sigma\Delta$  ADC with the frequency response of the first-order ADC, it is easily seen that the magnitude response at frequencies near zero becomes flatter the higher the order of the  $\Sigma\Delta$  converter that is chosen. At low frequencies, the sine function can be replaced by its argument; hence, for the first-order converter, NTF( $f$ ) varies with  $f$ , for the second order with  $f^2$  and for the  $n$ th-order with  $f^n$ . For higher-order  $\Sigma\Delta$  ADCs, the power density spectrum of the error sequence gets much more suppressed than for the first-order ADC; thus, much larger bit gains can be realized with relatively low oversampling ratios. The bit gain of the  $n$ th-order  $\Sigma\Delta$  ADC can be shown to be<sup>55</sup>

$$G = (n + 0.5) \text{ld(OSR)} + 0.5 \text{ld}(2n + 1) - n \text{ld}(\pi) \quad (7.16)$$

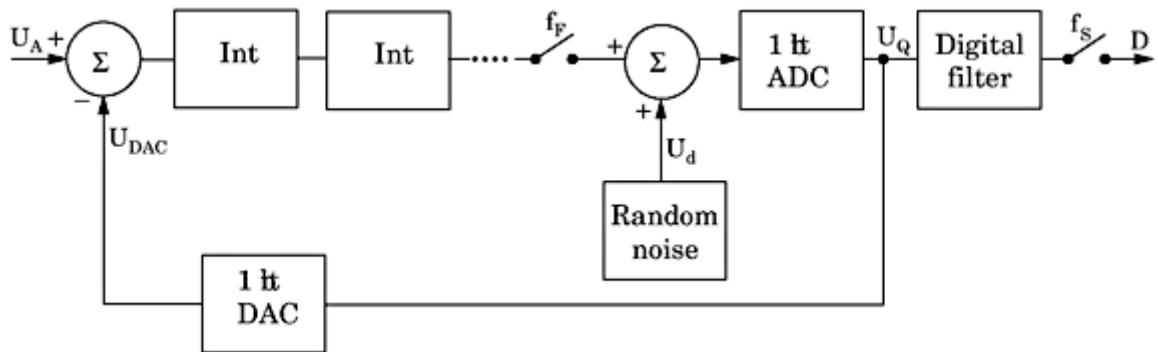
[Table 7.1](#) lists the bit gain for orders from 1 to 3 and for oversampling ratios up to 256.

We see that with a third-order  $\Sigma\Delta$  ADC, a bit gain of nearly 14 is obtained with an oversampling ratio as low as 32. Higher-order  $\Sigma\Delta$  ADCs seem to offer a very easy and elegant way to realize A/D converters with an extremely high resolution, still using the simplest ADC—in other words, the one-bit converter. But what about the randomness of the error sequence? Simulations and experiments with realized converters show that the error sequence  $e(nT)$  becomes more and more “random” the higher the order  $n$  is chosen. It turns out, however, that even at orders of 6 or higher, spurs are still visible in the spectrum. We will see in [Sec. 7.4.2](#) that such spurs become even more disturbing because in audio D/A applications they produce audible tones. To suppress spurs in higher-order  $\Sigma\Delta$  ADCs, a technique called *dithering* is applied.

**TABLE 7.1 The Bit Gain  $G$  of the  $n$ th-Order  $\Sigma\Delta$  ADC for  $n = 1$  to 3, and OSR Up to 256**

OSR	$G$ ( $n = 1$ )	$G$ ( $n = 2$ )	$G$ ( $n = 3$ )
8	3.64	5.36	6.95
16	5.14	7.86	10.45
32	6.64	10.36	13.95
64	8.14	12.86	17.45
128	9.64	15.36	20.95
256	11.14	17.86	24.45

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.



**Figure 7.11** A higher-order  $\Sigma\Delta$  ADC with dither signal  $U_d$

A  $\Sigma\Delta$  ADC that applies dithering is shown in Fig. 7.11. A random noise signal  $U_d$  is added to the input of the quantizer. This signal has uniform amplitude distribution in the range  $-Q/2$  to  $Q/2$ , where  $Q$  is the quantization step (in our examples,  $Q$  was chosen to be 1). Usually, a pseudo-random sequence is used for the dither signal.<sup>55</sup> Due to the added dither, the input signal of the first integrator becomes randomized as well, and the PSD of the quantization error becomes almost ideally flat in the frequency range from  $-f_F/2$  to  $f_F/2$ .

We did not mention another serious problem hitherto: the stability of the loop in higher-order  $\Sigma\Delta$  converters. It is well known that an integrator exhibits a phase shift of 90 degrees at higher frequencies. When two or more integrators are cascaded in a closed loop, the system becomes unstable. To get a stable system, the noise transfer function NTF( $f$ ) must be modified. Equation (7.14) shows that the NTF has only zeroes, not poles. The gain at the Nyquist frequency ( $f_F/2$ ) is  $2^n$  according to Eq. (7.15)—that is, it increases with higher system order. To get a stable system, poles must be added to the NTF, which results in the modified noise transfer function

$$\text{NTF}(z) = \frac{(1 - z^{-1})^n}{P_n(z^{-1})} \quad (7.17)$$

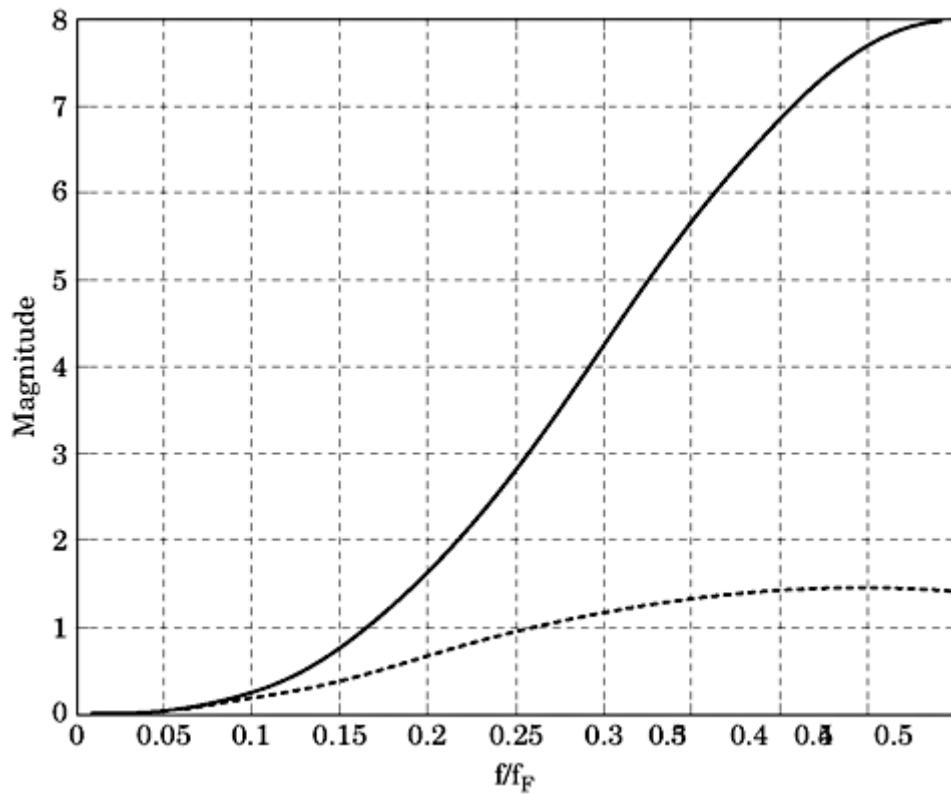
where  $P_n(z^{-1})$  is an  $n$ th-order polynomial in  $z^{-1}$ . This lowers the noise gain at the Nyquist frequency (cf. Fig. 7.12). The NTF of a third-order system is shown here. Without additional poles, the gain is 8 at the Nyquist frequency. With additional poles, the gain is reduced to 1.4. It is important to note that the poles have a strong influence on the frequency response at higher frequencies but do not significantly alter the frequency response at lower frequencies, say, near zero. The bit gain [cf. Eq. (7.16)] therefore remains nearly unchanged.

Kuo and coauthors<sup>56</sup> have discovered through simulations that the noise power gain (NPG) is the relevant parameter for loop stability. Noise power gain is defined by

$$\text{NPG} = \frac{1}{f_F} \int_{-f_F/2}^{f_F/2} |\text{NTF}(f)|^2 df \quad (7.18)$$

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.



**Figure 7.12** Noise transfer functions of a third-order  $\Sigma\Delta$  ADC. Solid curve: without compensation. Dotted curve: with compensation by additional poles.

When the oversampling ratio is high (which is the case in most applications) NPG is approximately equal to the squared noise gain at the Nyquist frequency

$$\text{NPG} \approx |\text{NTF}(f_F/2)|^2 \quad (7.19)$$

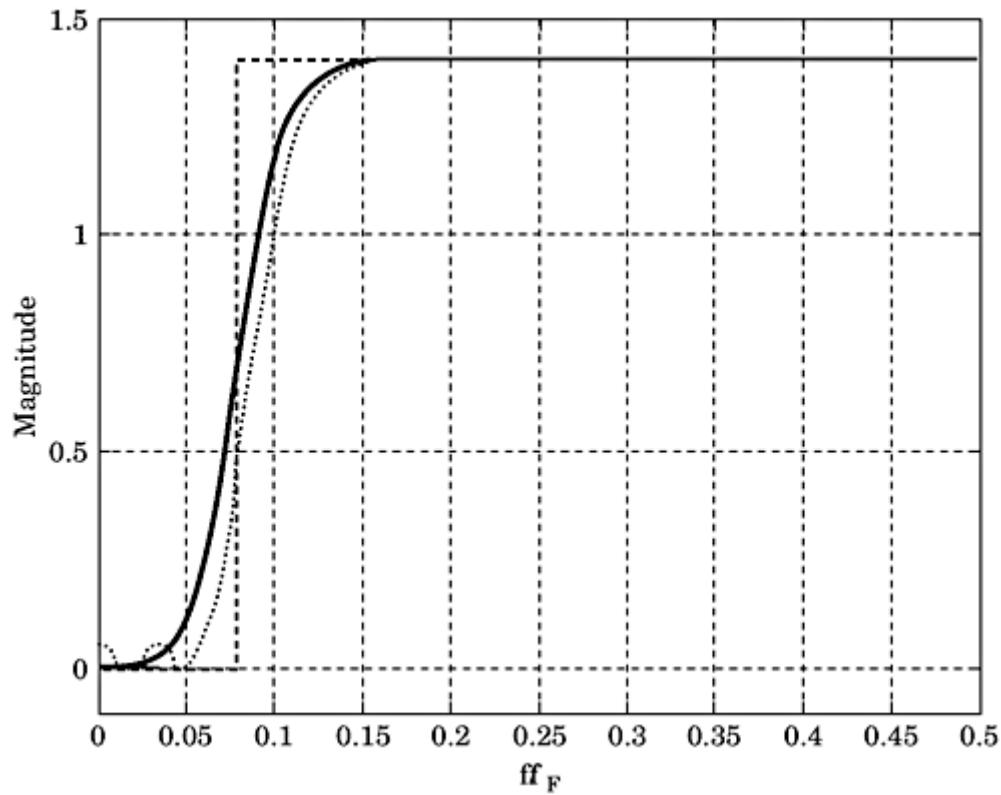
The simulations performed by Kuo *et al.* have shown that the system stays stable whenever the noise power gain is less than a maximum  $\text{NPG}_{\max}$ . They found an empirical formula for the maximum noise power gain:

$$2.55 - 3\frac{U_{A,\max}^2}{2}, \quad n = 3 \quad (7.20)$$

$$\text{NPG}_{\max} = 2.4 - 3\frac{U_{A,\max}^2}{2}, \quad n = 4$$

$$2.35 - 3\frac{U_{A,\max}^2}{2}, \quad n = 5 \dots 8$$

where  $U_{A,\max}$  is the maximum analog input signal, which is 1 in our application. We must now determine the location of the poles. A suitable procedure has been presented by Kuo.<sup>56</sup> Given the required order  $n$  and the OSR of the  $\Sigma\Delta$  ADC, the NTF is predefined as a highpass filter function. Because the NTF must suppress



**Figure 7.13** Defining the NTF of the  $\Sigma\Delta$  ADC using a highpass filter function.

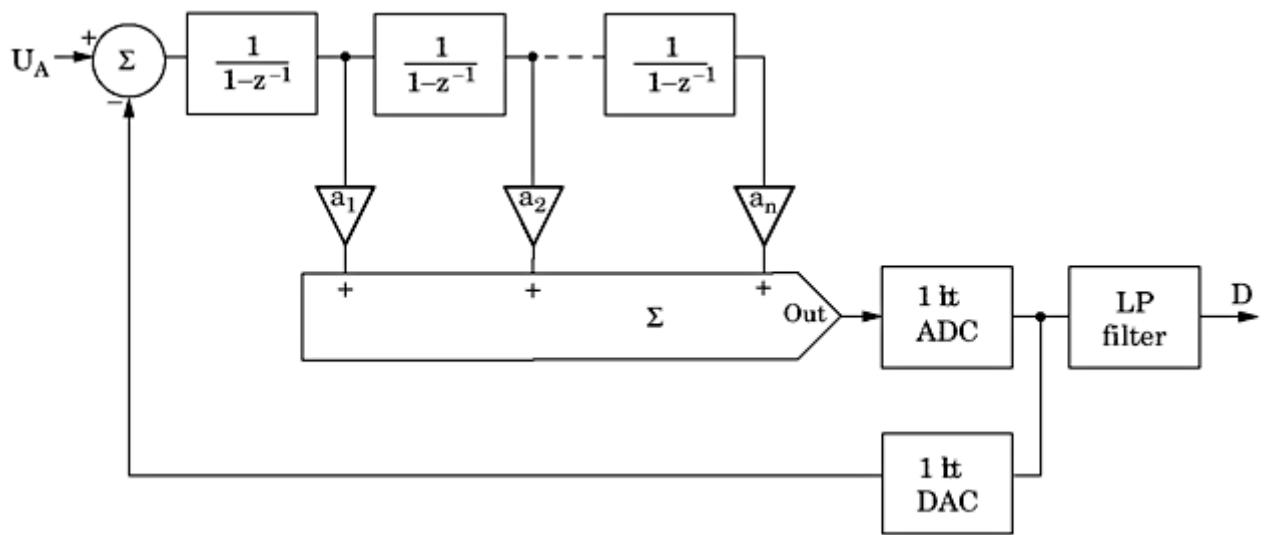
the frequencies in the range 0 to  $f_S/2$  (with  $f_F/f_S = \text{OSR}$ ), the stopband-edge frequency of this filter is set

$$f_c = \frac{f_F/2}{\text{OSR}} \quad (7.21)$$

If the highpass filter were ideal, its frequency response would be represented by the dashed curve in Fig. 7.13. Two types of IIR filters lend themselves for the practical implementation: the Butterworth and the inverse Chebyshev filter (also called Chebyshev Type 2).<sup>57</sup> The frequency response of the Butterworth highpass is represented by the solid curve, and the Chebyshev 2 filter by the dotted curve. The Butterworth highpass has all its zeroes at  $s = 0$ ; hence, the magnitude response is optimally flat around  $f = 0$ . With the Chebyshev 2 filter, the zeroes are distributed across frequencies in the range  $-f_c$  to  $f_c$ . This offers the advantage that the filter attenuation of the Chebyshev 2 filter is larger within the stopband than in the case of the Butterworth filter.

To complete the filter specification, we must determine the filter gain at the Nyquist frequency. This can be done by first specifying the maximum allowable noise power gain [Eq. (7.20)] and using the approximation of Eq. (7.19) to compute  $\text{NTF}(f_F/2)$ . A conventional filter design program can then be used to design the highpass filter (for example, Matlab<sup>25</sup>). Usually such programs design filters whose frequency response is 1 at the Nyquist frequency. Because

the gain at the Nyquist frequency is larger in our application (mostly in the range 1.2 to 1.6), we must scale the numerator coefficients correspondingly, which is done by multiplying all numerator coefficients by  $\text{NTF}(f_F/2)$ .



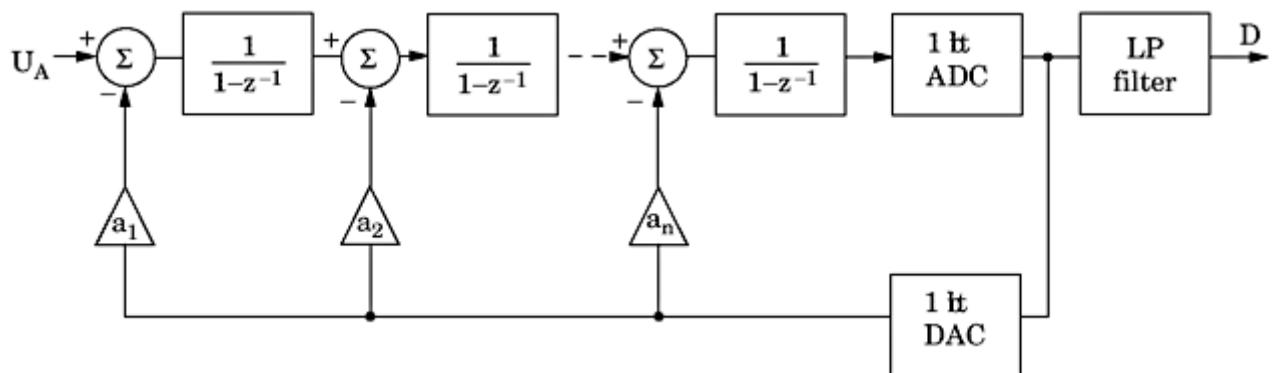
**Figure 7.14** An  $n$ th-order  $\Sigma\Delta$  ADC with weighted feedforward summation.

Knowing the noise transfer function of the  $\Sigma\Delta$  ADC, we must now check how the block diagram in Fig. 7.11 should be modified in order to implement the required NTF. This can be done in several ways. Fig. 7.14 shows an  $n$ th-order  $\Sigma\Delta$  ADC with weighted feedforward summation. The output of each integrator is scaled by weights  $a_1$  to  $a_n$ , and the sum of the scaled integrator outputs is fed to the input of the quantizer. The  $NTF(z)$  of this configuration now becomes

$$NTF_{FF}(z) = \frac{(1 - z^{-1})^n}{(1 - z^{-1})^n + a_1(1 - z^{-1})^{n-1} + \dots + a_n} \quad (7.22)$$

When comparing this expression with the NTF of a  $\Sigma\Delta$  ADC with no feedforward compensation [cf. Eq. (7.14)], it shows up that this noise transfer function has zeroes and poles. The zeroes remain the same as in Eq. (7.14); hence, the frequency response at frequencies near zero is almost unchanged. The poles, however, will decrease the magnitude at higher frequencies in order to enable stable operation.

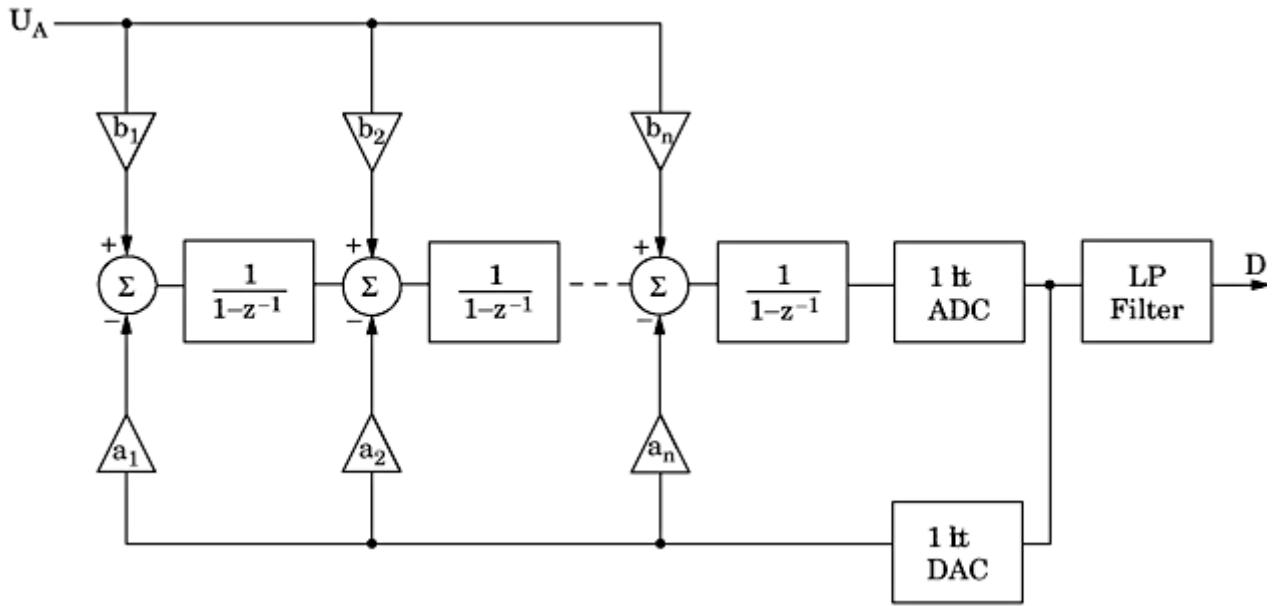
A couple of other compensation schemes can be used. Two of them are plotted in the next two figures. Figure 7.15 is an  $n$ th-order  $\Sigma\Delta$  ADC with feedback



**Figure 7.15** An  $n$ th-order  $\Sigma\Delta$  ADC with weighted feedback paths.

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.



**Figure 7.16** An  $n$ th-order  $\Sigma\Delta$  ADC with weighted feedforward and weighted feedback paths.

compensation. Here, the output signal of the one-bit DAC is not only fed to the input of the first integrator but to all integrators using weights  $a_1$  through  $a_n$ .

Another way to obtain stable operation is to combine both feedforward and feedback paths, as shown in Fig. 7.16. The signals fed back are scaled by weights  $a_1$  to  $a_n$ , whereas the feed forward signals are scaled by weights  $b_1$  to  $b_n$ .

When one of these configurations has been chosen for the  $n$ th-order  $\Sigma\Delta$  ADC, the scaling factors  $a_i$  and/or  $b_i$  must be computed in order to obtain the required frequency response NTF( $z$ ). Determination of scaling factors will be shown by the example of the feedforward configuration in Fig. 7.14. We assume the designer has predefined a suitable NTF( $z$ ), by selecting either a Butterworth or a Chebishev 2 highpass function. By using a digital filter design program, the designer will have obtained the NTF( $z$ ) in the form

$$\text{NTF}(z) = \frac{d_0 + d_1 z^{-1} + \dots + d_n z^{-n}}{1 + c_1 z^{-1} + \dots + c_n z^{-n}} \quad (7.23)$$

where the  $c_i$  and  $d_i$  are the filter coefficients determined by the filter design program. This expression must now be equal with NTF<sub>FF</sub>( $z$ ), as given in Eq. (7.22), which is

$$\frac{d_0 + d_1 z^{-1} + \dots + d_n z^{-n}}{1 + c_1 z^{-1} + \dots + c_n z^{-n}} = \frac{(1 - z^{-1})^n}{(1 - z^{-1})^n + a_1(1 - z^{-1})^{n-1} + \dots + a_n} \quad (7.24)$$

Because the coefficients of all the powers in  $z$  in the denominator must be the same on both sides of the equation, this yields  $n$  equations to determine the  $a_i$  weights. (Note that the

numerators can only be identical when a Butterworth filter has been chosen to implement  $NTF(z)$ . Otherwise, another structure would

have to be chosen for the  $\Sigma\Delta$  ADC, say, the configuration with both feedback and feedforward paths [cf. Fig. 7.16].) A numerical example of a fourth-order  $\Sigma\Delta$  ADC with weighted feedword summation is given in Norsworthy and coauthors.<sup>55</sup>

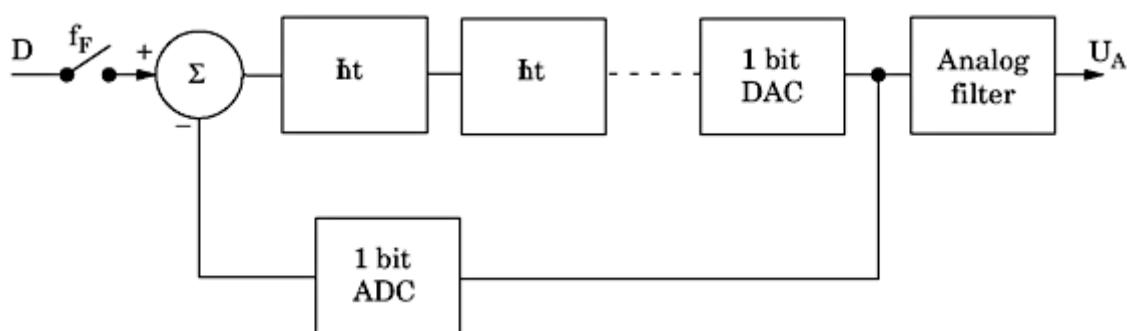
As mentioned earlier, the error sequence of the quantizer is processed by two filters in cascade. The first of these is the highpass filter with frequency response  $NTF(f)$  that rejects low frequencies between 0 and the stopband-edge frequency. Frequencies above the stopband-edge pass through the filter and thus must be removed by the postprocessing digital lowpass filter, as shown in Figs. 7.14, 7.15, and 7.16. To efficiently reject those higher frequencies, the cutoff frequency of the digital filter should be chosen to coincide with the stopband-edge frequency of the highpass filter. The lowpass filter is usually implemented as an FIR filter. It is sampled at the fast frequency  $f_F$  but the output signal  $D$  is read out at the slow sampling rate  $f_S$ , which is lower than  $f_F$  by the factor OSR. Because the transition region of the lowpass filter should be narrow in order to sufficiently reject the frequencies above  $f_S/2$ , the length (order) of the FIR filter must be chosen to be large—usually around 100.

Every  $\Sigma\Delta$  ADC consists of two parts: an analog portion and a digital one. The integrators are operating in continuous time, and hence are linear circuits. They can be implemented either as active RC integrators or as switched capacitor filters (SC).

### The $\Sigma\Delta$ D/A converter

One of the first applications of the  $\Sigma\Delta$  DAC was in audio CD players. The audio signal on the CD is encoded with a word length of 16 bits. Data are read out with a sampling rate of 44.1 kHz, which is twice the bandwidth of the audio signal. To play the CD on a linear amplifier, the digital signal must be converted to analog. Because 16-bit DACs have proven rather expensive, manufacturers of CD players used  $\Sigma\Delta$  DACs built from an extremely simple one-bit DAC. Figure 7.17 shows the simplified block diagram of an  $n$ th-order  $\Sigma\Delta$  DAC. The digital input signal  $D$  is sampled with rate  $f_F$ , which is higher by a factor OSR than the initial sampling rate  $f_S$  of the original signal.

Assume for the moment that  $f_S = 44.1$  kHz and that we are going to over-sample that signal by  $OSR = 256$ . The input sampling rate of the DAC therefore becomes  $256 \cdot 44.1$  kHz = 11.289 MHz. When the order of the  $\Sigma\Delta$  DAC is

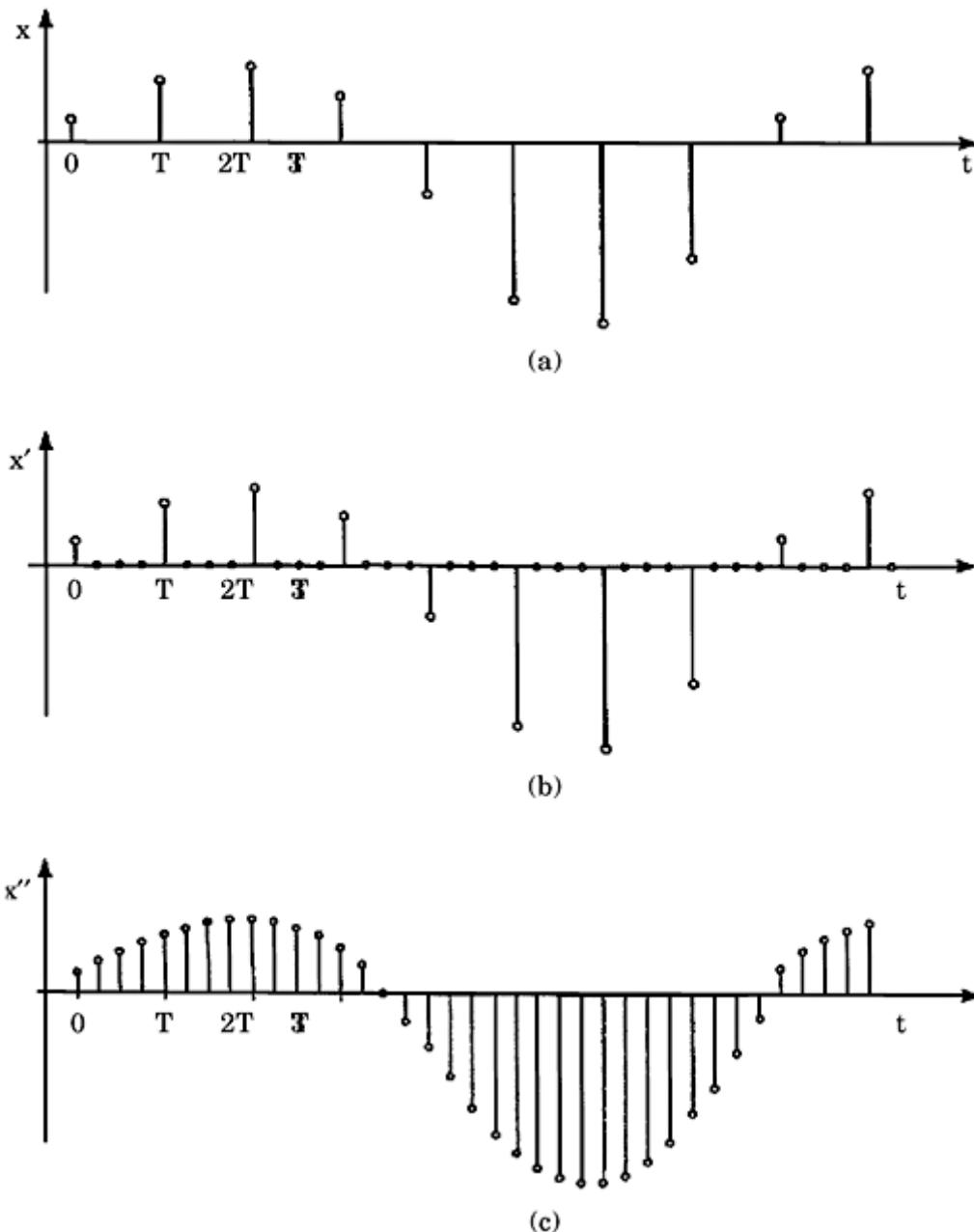


**Figure 7.17** A simplified block diagram of an  $n$ th-order  $\Sigma\Delta$  DAC.

**Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.**

chosen as  $n = 2$ , this oversampling ratio yields a bit gain  $G = 17.86$  (from [Table 7.1](#)). Upsampling a digital signal is usually accomplished by interpolation. This is explained in [Fig. 7.18](#). [Figure 7.18a](#) is the original digital signal, sampled at  $f_S = 1/T$ . To upsample the signal by a factor of 4, for example, we insert three zeroes between any two succeeding signal samples, as shown in [Fig. 7.18b](#). This technique is also called *zero padding*. Now the sampling rate is 4  $f_S$ . Additional samples are interpolated by lowpass filtering the zero padded signal with cutoff frequency  $f_S/2$ . A digital lowpass filter is used for this purpose.

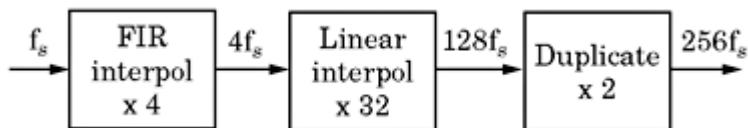
Now we must upsample our signal by 256 and not by 4; hence, we must insert 255 zeroes between any two successive samples, and then lowpass filter the zero



**Figure 7.18** Upsampling a digital signal by zero padding and lowpass filtering. (a) The original digital signal sampled at  $f_S = 1/T$ . (b) Zeroes are inserted between any two succeeding samples. (c) Lowpass filtering the zero-padded signal with cutoff frequency  $f_S/2$  yields the interpolated signal, upsampled to  $4f_S$ .

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.



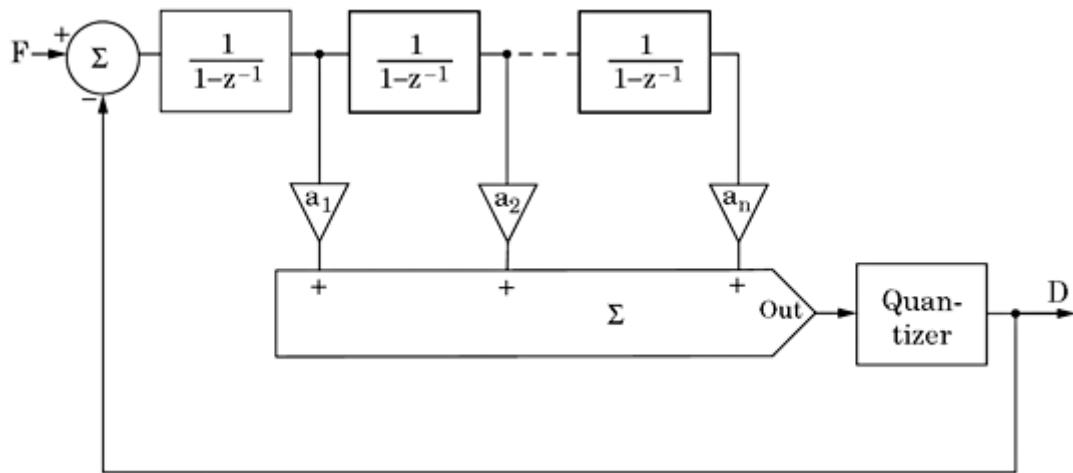
**Figure 7.19** Upsampling a digital signal by a factor 256. First block: upsampling x4 by zero padding and lowpass filtering. Second block: linear interpolation x32. Third block: duplicating samples x2.

padded signal. When using a FIR lowpass filter, this should operate at around 11.3 MHz, which does not seem unrealistic. However, the filter should have a cutoff frequency of only 22 kHz with a relatively sharp transition region, say, 1 kHz wide at most. Such a digital filter would require a length on the order of at least 10,000, and the filter algorithm would have to perform at least  $11.3 \cdot 10^6 \cdot 10,000$  MACs (multiply-accumulate) operations per second, which is more than 100 Giga MACs/s.

This by far exceeds the capabilities of presently available microprocessors or DSPs, so we have to look for a simpler method of upsampling. [Figure 7.19](#) shows an upsampling algorithm used in the integrated circuit type SAA 7320, manufactured by Philips.<sup>59</sup> First, the digital input signal is interpolated by a factor of 4 by zero padding and lowpass filtering using a FIR filter. This filter has 128 taps. After the FIR filter, the sampling rate is 176 kHz. Next, the upsampled signal is linearly interpolated by a factor of 32; thus, a microprocessor simply calculates 31 additional samples between every sample pair by linear interpolation. This is not fully correct in a mathematical sense but the errors are so small they can be neglected. After the second stage, the signal is upsampled by a factor of 128. The third block just duplicates every sample; thus, the final oversampling factor is 256. The IC uses a two-stage  $\Sigma\Delta$  DAC for signal processing. The analog output signal is finally lowpass-filtered by a three-stage switched capacitor filter. The SAA 7320 IC uses a dither signal to eliminate audible tones.

## The $\Sigma\Delta$ modulator used in frequency synthesizers

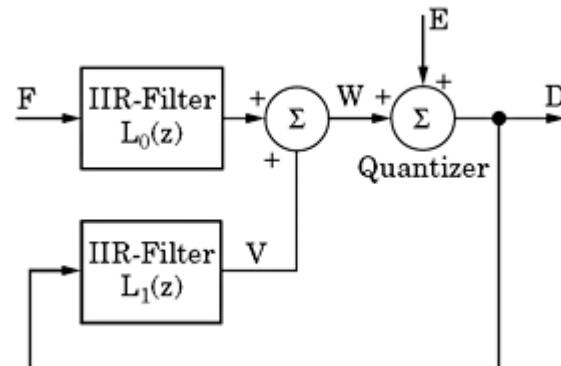
In this section, we are dealing with the design of  $\Sigma\Delta$  modulators used in fractional-*N* frequency synthesizers with digital spur suppression. The block diagram of such a system has been shown in [Fig. 7.4](#). In this application, the  $\Sigma\Delta$  modulator converts a given fractional number *F* into binary signal *D*—in other words, into a sequence of 1s and 0s whose average value equals the fractional number *F*. The structure of the  $\Sigma\Delta$  modulator can be derived immediately from structures of known  $\Sigma\Delta$  A/D converters. [Figure 7.20](#) shows one possible  $\Sigma\Delta$  modulator structure, which has the same topology as the  $\Sigma\Delta$  ADC in [Fig. 7.14](#) (weighted feedforward summation). There are, however, some minor but important differences. In the  $\Sigma\Delta$  modulator, all signals are digital. The integrators are digital integrators, of course, and the one-bit ADC in [Fig. 7.14](#) is replaced by a digital quantizer having a transfer characteristic as plotted in [Fig. 7.5b](#). Because the input signal of the  $\Sigma\Delta$  modulator (*F*) is digital, the one-bit DAC (cf. 7.14) is no longer needed. We recognize, furthermore, that the digital lowpass filter



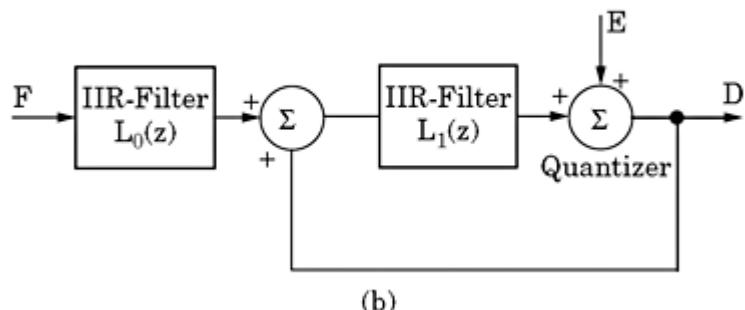
**Figure 7.20** A simplified block diagram of an  $n$ th-order  $\Sigma\Delta$  modulator built from a chain of integrators and distributed feedforward paths.

(Fig. 7.14) has been omitted in the block diagram of Fig. 7.20. As will be demonstrated later, the function of this lowpass filter is taken over by the transfer function  $H(f)$  of the PLL.

Any other structure could be taken to build a  $\Sigma\Delta$  modulator, such as the structures shown in Figs. 7.15 and 7.16. We will recognize, however, that there are more convenient design procedures—for example, those based on the block diagrams in Fig. 7.21a and b.



(a)



(b)

**Figure 7.21** A generalized block diagram of a  $\Sigma\Delta$  modulator built from two IIR filters. The

quantizer has been replaced by its mathematical model. (a) Structure 1: Can be realized only if  $\text{NTF}(z)$  does not have a zero at  $z = 1$ . (b) Structure 2: Can be used for any  $\text{NTF}(z)$ .

Two of many possible structures are shown here. Let's start with the first of these (cf. Fig. 7.21a). The dynamic performance of this  $\Sigma\Delta$  modulator is determined by two IIR filters having transfer functions  $L_0(z)$  and  $L_1(z)$ , respectively. In this figure, the quantizer has been replaced by its mathematical model (cf. Fig. 7.5c). From this block diagram, the following relations can be derived:

$$W(z) = FL_0(z) + DL_1(z) \quad (7.25a)$$

$$D(z) = W(z) + E(z) \quad (7.25b)$$

Substituting Eq. (7.25b) into (7.25a) yields (after some manipulations)

$$D(z) = F(z) \frac{L_0(z)}{1 - L_1(z)} + E(z) \frac{1}{1 - L_1(z)} \quad (7.26)$$

This can be rewritten as

$$D(z) = F(z) \cdot \text{STF}(z) + E(z) \cdot \text{NTF}(z) \quad (7.27)$$

with  $\text{STF}(z)$  = signal transfer function, and  $\text{NTF}(z)$  = noise transfer function. By comparing Eqs. (7.26) and (7.27), we get

$$L_0(z) = \frac{\text{STF}(z)}{\text{NTF}(z)} \quad (7.28a)$$

$$L_1(z) = \frac{\text{NTF}(z) - 1}{\text{NTF}(z)} \quad (7.28b)$$

These equations reveal a great advantage of the new configuration: both signal transfer function  $\text{STF}(z)$  and noise transfer function  $\text{NTF}(z)$  can be specified independently of each other. As explained by Norsworthy and coauthors,<sup>55</sup> this is not possible with the configurations of Figs. 7.14 through 7.16; here, only one transfer function can be prespecified, normally  $\text{NTF}(z)$ . The  $\text{STF}(z)$  then depends on the selected  $\text{STF}(z)$  and sometimes exhibits adverse peaking effects. There is an important restriction, however, on the specification of the noise transfer function  $\text{NTF}(f)$ . When a suitable highpass filter function  $\text{NTF}(z)$  has been designed, this function is available in the form

$$\text{NTF}(z) = \frac{\text{num}(z)}{\text{den}(z)} = \frac{b_0 + b_1 z^{-1} + \dots + b_n z^{-n}}{1 + a_1 z^{-1} + \dots + a_n z^{-n}} \quad (7.29)$$

where  $\text{num}(z)$  and  $\text{den}(z)$  are the numerator and denominator polynomials, respectively, and  $a_i$  and  $b_i$  are the filter coefficients. The transfer function of the filter  $L_1(z)$  then becomes

$$L_1(z) = \frac{\text{num}(z) - \text{den}(z)}{\text{num}(z)} = \frac{d_0 + d_1 z^{-1} + \dots + d_n z^{-n}}{1 + c_1 z^{-1} + \dots + c_n z^{-n}} \quad (7.30a)$$

where the  $c_i$  and  $d_i$  are the coefficients of filter  $L_1(z)$ . We now observe that this IIR operates in a closed loop—in other words, its output signal  $v(nT)$  is immediately fed back (via the quantizer) to its input without a delay. Therefore, the output signal  $v(nT)$  in the  $n$ th sampling interval cannot depend on the input signal in the same interval, and consequently the constant coefficient  $d_0$  in the numerator of  $L_1(z)$  must be zero. This implies that the constant coefficient  $b_0$  in the numerator of  $\text{NTF}(z)$  must be 1. This must be taken into account when designing the highpass function  $\text{NTF}(z)$ . As stated earlier, the signal transfer function can be chosen arbitrarily, so it seems convenient to choose  $\text{STF}(z) = 1$ . When doing so, the transfer function  $L_0(z)$  becomes

$$L_0(z) = \frac{1}{\text{NTF}(z)} = \frac{\text{den}(z)}{\text{num}(z)}$$

[cf. [Eq. \(7.28a\)](#)]. When a Butterworth highpass function has been chosen for  $\text{NTF}(z)$ , the noise transfer function has one or more *zeroes* at  $z = 1$ . This implies that  $L_0(z)$  has one or more *poles* at  $z = 1$ , which means that the input signal  $F$  passes through one or more *integrators*. Because the filter  $L_0(z)$  operates in open loop, its output signal runs away toward infinity for any nonzero input signal  $F$ . We conclude that the structure in [Fig. 7.21a](#) can only be realized when the noise transfer function does not have zeroes at  $z = 1$ , which is fulfilled exclusively for Chebyshev 2 filters of even order.

This problem can be fixed by choosing a different filter structure, as shown in [Fig. 7.21b](#). For this configuration, the output signal  $D(z)$  is given by

$$D(z) = F(z) \frac{L_0(z)L_1(z)}{1 - L_1(z)} + E(z) \frac{1}{1 - L_1(z)}$$

Hence, the signal and noise transfer functions become

$$\text{STF}(z) = \frac{L_0(z)L_1(z)}{1 - L_1(z)}$$

$$\text{NTF}(z) = \frac{1}{1 - L_1(z)}$$

As done previously, we start by defining the noise transfer function by

$$\text{NFT}(z) = \frac{\text{num}(z)}{\text{den}(z)} = \frac{b_0 + b_1 z^{-1} + \dots + b_n z^{-n}}{1 + a_1 z^{-1} + \dots + a_n z^{-n}}$$

[cf. [Eq. \(7.29\)](#)]. For the transfer function  $L_1(z)$ , we again get

$$L_1(z) = \frac{\text{num}(z) - \text{den}(z)}{\text{num}(z)} = \frac{d_0 + d_1 z^{-1} + \dots + d_n z^{-n}}{1 + c_1 z^{-1} + \dots + c_n z^{-n}}$$

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

which is same as that obtained for the first structure. To get a realizable filter,  $d_0$  must become 0 again, which implies that the numerator coefficient  $b_0$  must be 1. Let's now consider the other filter function  $L_0(z)$ . For the second structure,  $L_0(z)$  can be shown to be

$$L_0(z) = \text{STF}(z) \frac{1 - L_1(z)}{L_1(z)}$$

Applying Eq. (7.28b) we get for  $L_0(z)$

$$\begin{aligned} L_0(z) &= \text{STF}(z) \frac{\text{den}(z)}{\text{num}(z) - \text{den}(z)} \\ &= \text{STF}(z) \frac{1 + a_1 z^{-1} + \dots}{(b_0 - 1) + (b_1 - a_1)z^{-1} + \dots} \end{aligned} \quad (7.30b)$$

Knowing that  $b_0 = 1$ , we recognize that the constant term in the denominator of Eq. (7.30b) becomes 0. If  $\text{STF}(z)$  were chosen to be 1,  $L_0(z)$  would be the transfer function of a noncausal filter; the output signal at sampling instant  $t = nT$  would depend on an input sample at  $t = (n + 1)T$ , which is impossible to realize in real time. The filter gets causal when setting  $\text{STF}(z) = z^{-1}$  (or  $z^{-2}, z^{-3}$  to  $z^{-k}$ ). Choosing  $\text{STF}(z) = z^{-1}$ , we obtain

$$L_0(z) = \frac{1 + a_1 z^{-1} + \dots}{(b_1 - a_1) + \dots (b_2 - a_2)z^{-1} + \dots}$$

which is the transfer function of a causal filter. We conclude therefore that the structure in Fig. 7.21b should be chosen whenever the  $\text{NTF}(z)$  has one or more zeroes at  $z = 1$ , which is the case for Butterworth highpass filters and for Chebyshev 2 highpass filters of odd order.

We have seen that switching the divider ratio of a down scaler (cf. Fig. 7.4) from a value  $N$  to another value  $N + 1$  inevitably creates some phase noise in the PLL system. Next, we will investigate what kind of phase noise at the output of the frequency synthesizer (output of the VCO) should be expected for a given  $\Sigma\Delta$  modulator configuration. For the following analysis, the mathematical model in Fig. 7.22 is used. In this block diagram, a number of symbols is introduced. Let's start with the symbols relating to the  $\Sigma\Delta$  modulator. The input of this modulator is the fractional part  $f$  of the divider ratio, which is  $N + f$ , where  $N$  is the integer part. Assuming that the  $\text{STF}(f)$  of the  $\Sigma\Delta$  modulator has been specified to be 1, the output of the  $\Sigma\Delta$  modulator is

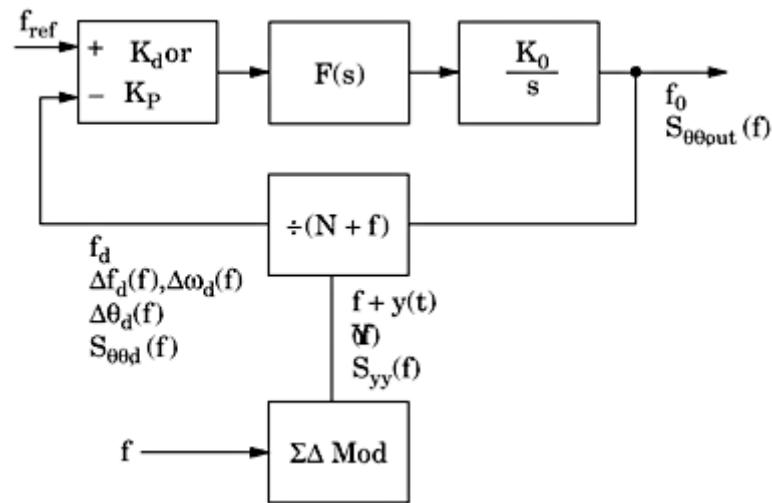
$$D(z) = f + \text{NTF}(z) \cdot E(z) \quad (7.31)$$

[refer to Eq. (7.27)]. When transformed back into the time domain, the output signal of the  $\Sigma\Delta$  modulator is

$$d(t) = f + \zeta^{-1}[\text{NTF}(z) \cdot E(z)] \quad (7.32)$$

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.



**Figure 7.22** A mathematical model of a fractional- $N$  frequency synthesizer for the computation of the PSD of phase noise created by the  $\Sigma\Delta$  modulator.

where  $\zeta^{-1}$  signifies the inverse  $z$  transform. The output signal  $d(t)$  is therefore the sum of the fractional part  $f$  plus the highpass-filtered error sequence  $e(nT)$ . For the second term in Eq. (7.32), we will use the symbol  $y(t)$ . The spectrum of  $y(t)$  is denoted to be  $Y(f)$ , and its power spectral density is  $S_{yy}(f)$ .

Provided the error sequence  $e(nT)$  is sufficiently randomized, it is a uniformly distributed sequence in the range  $-Q/2$  to  $Q/2$ , with  $Q = 1$ . Hence, its variance is given by

$$\text{Var}[e(nT)] = \frac{Q^2}{12} \quad (7.33)$$

If we interpret the error sequence  $e(nT)$  as a voltage signal applied to a load resistor of 1 Ohm,  $\text{Var}[e(nT)]$  is identical with the signal power. Because the error sequence is assumed to be fully random, its power spectral density is constant within the frequency interval  $-f_{\text{ref}}/2$  to  $f_{\text{ref}}/2$ . The PSD of the error sequence then becomes

$$S_{ee}(f) = \frac{Q^2}{12f_{\text{ref}}} \quad [\text{W/Hz}] \quad (7.34)$$

Now the PSD of the highpass-filtered error sequence  $y(t)$  can be computed from

$$S_{yy}(f) = S_{ee}(f)|\text{NTF}(f)|^2 = \frac{Q^2}{12f_{\text{ref}}} \cdot |\text{NTF}(f)|^2 \quad (7.35)$$

For the absolute value of the frequency spectrum of  $y(t)$ , we can therefore write

$$|Y(f)| = \sqrt{S_{yy}(f)} = \frac{Q}{\sqrt{12f_{ref}}} \cdot |\text{NTF}(f)| \quad (7.36)$$

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

We will use the spectrum  $Y(f)$  to compute the phase noise at the output of the down scaler in Fig. 7.22. The division ratio of the down scaler is  $N + f$  with  $N$  = integer part and  $f$  = fractional part. Due to the error  $y(t)$ , the frequency at the output of the down scaler is no longer constant but is modulated by the  $y$  signal. The output frequency of the down scaler  $f_d(t)$  consequently becomes

$$f_d(t) = \frac{f_0}{N + f + y(t)}$$

or, when setting  $N + f = M$  (overall division ratio), we have

$$f_d(t) = \frac{f_0}{M + y(t)} \quad (7.37)$$

Because  $f_0/M$  equals the reference frequency  $f_{\text{ref}}$ , this can be rewritten as

$$f_d(t) = \frac{f_0}{M \left(1 + \frac{y(t)}{M}\right)} = \frac{f_{\text{ref}}}{1 + \frac{y(t)}{M}} \quad (7.38)$$

Now  $y(t)$  is much less than  $M$ , hence we can use the approximation  $\frac{1}{1 + \varepsilon} \approx 1 - \varepsilon$  and get

$$f_d(t) = f_{\text{ref}} \left(1 - \frac{y(t)}{M}\right) \quad (7.39)$$

The deviation of  $f_d$  from its nominal value  $f_{\text{ref}}$  is denoted as  $\Delta f_d(t)$ , which is given by

$$\Delta f_d(t) = -\frac{y(t)}{M} f_{\text{ref}} \quad (7.40)$$

Now we don't know explicitly the highpass-filtered error sequence  $y(t)$ , but we know its spectrum from Eq. (7.36) so we can compute the spectrum of the frequency deviation, which will be denoted as  $\Delta F_d(f)$ . At frequency  $f$ , the spectrum of the frequency deviation becomes

$$\Delta F_d(f) = \frac{Y(f)}{M} f_{\text{ref}} \quad (7.41a)$$

If there existed only this spectral line at frequency  $f$ , the frequency deviation versus time  $\Delta f_d(t)$  would be

$$\Delta f_d(t) = \frac{Y(f)\sqrt{2} \cos(2\pi ft)}{M} f_{\text{ref}} \quad (7.41b)$$

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

The factor  $\sqrt{2}$  is inserted because  $Y(f)$  is the rms value of the spectrum amplitude at frequency  $f$ . Next, we calculate the deviation of radian frequency versus time at frequency  $f$  and get

$$\Delta\omega_d(t) = \frac{2\pi Y(f)\sqrt{2}\cos(2\pi ft)}{M} f_{\text{ref}} \quad (7.42)$$

Knowing the deviation of radian frequency, we can compute the phase deviation  $\Sigma\theta_d(t)$  of  $f_d(t)$  at frequency  $f$  from

$$\Delta\theta_d(t) = \int_0^t \Delta\omega_d(t) dt = \frac{Y(f)\sqrt{2}\sin(2\pi ft)}{fM} f_{\text{ref}} \quad (7.43)$$

Next, the power spectral density of the phase deviation is computed, which is simply the square of the rms value of the amplitude of  $\Delta\theta_d(f)$ . We will denote this quantity by  $L_{\theta\theta,d}(f)$

$$L_{\theta\theta,d}(f) = \frac{|Y(f)|^2 f_{\text{ref}}^2}{f^2 M^2} \quad [\text{rad}^2/\text{Hz}] \quad (7.44)$$

$L_{\theta\theta,d}(f)$  is the PDS of phase perturbation  $\Delta\theta_d(t)$  at the output of the down scaler at frequency  $f$ . But the spectrum has an identical line at frequency  $-f$ , hence the PSD of the one-sided spectrum is twice as large. It is denoted by  $S_{\theta\theta,d}(f)$  and is given by

$$S_{\theta\theta,d}(f) = \frac{2|Y(f)|^2 f_{\text{ref}}^2}{f^2 M^2} \quad [\text{rad}^2/\text{Hz}] \quad (7.45)$$

The phase perturbation  $\Delta\theta_d(t)$  appears at the input of the phase detector (cf. [Fig. 7.22](#)). It is therefore processed by the PLL in the same way as the phase noise of the reference oscillator, which was discussed in [Sec. 6.7.1](#). Using [Eq. \(6.16\)](#), the contribution of phase perturbation  $S_{\theta\theta,d}(f)$  to the phase perturbation  $S_{\theta\theta,\text{out}}(f)$  at the output of the VCO then becomes

$$S_{\theta\theta,\text{out}}(f) = S_{\theta\theta,d}(f) \cdot M^2 \cdot |H(f)|^2 \quad [\text{rad}^2/\text{Hz}] \quad (7.46)$$

Using Eqs. [\(7.35\)](#), [\(7.36\)](#), and [\(7.45\)](#), we get

$$S_{\theta\theta,\text{out}}(f) = \frac{2f_{\text{ref}}^2}{f^2} |H(f)|^2 \frac{Q^2}{12f_{\text{ref}}} |\text{NTF}(f)|^2 \quad [\text{rad}^2/\text{Hz}] \quad (7.47)$$

We assume for the moment that a Butterworth highpass filter is used for  $\text{NTF}(f)$ ; its transfer function has been given in [Eq. \(7.9b\)](#). For the squared absolute value  $|\text{NTF}(f)|$ , we therefore

get

$$|\text{NTF}(f)|^2 = 2^{2n} \sin\left(\frac{\pi f}{f_{\text{ref}}}\right)^{2n} \quad (7.48)$$

and the PSD of phase perturbation at the output of the VCO becomes

$$S_{\theta\theta,\text{out}}(f) = \frac{2f_{\text{ref}}^2}{f^2} |H(f)|^2 \frac{Q^2}{12f_{\text{ref}}} 2^{2n} \sin\left(\frac{\pi f}{f_{\text{ref}}}\right)^{2n} [\text{rad}^2/\text{Hz}] \quad (7.49)$$

The highpass filter  $\text{NTF}(f)$  suppresses the lower frequencies—in other words, the frequencies between 0 and the stopband-edge. At those frequencies, the phase transfer function  $H(f)$  is approximately 1, and for low frequencies we can replace the sine function by its argument. Thus, for low frequencies the PSD of phase perturbation is approximated by ( $Q = 1$ )

$$S_{\theta\theta,\text{out}}(f) = \frac{2 \cdot 2^{2n} \pi^{2n}}{12f_{\text{ref}}} \cdot \frac{f^{2n-2}}{f_{\text{ref}}^{2n-2}} [\text{rad}^2/\text{Hz}] \quad (7.50)$$

We conclude that phase noise contributed by the  $\Sigma\Delta$  modulator varies with  $f^{2n-2}$  at low frequencies. Consequently, there is no noise suppression for a first-order  $\Sigma\Delta$  modulator. For the second-order  $\Sigma\Delta$  modulator, the phase noise varies with  $f^2$ , for the third-order  $\Sigma\Delta$  modulator with  $f^4$ , and so on. The noise suppression therefore becomes better the higher the order  $n$  is chosen. At frequencies above the stopband-edge of the highpass filter, the phase noise increases very rapidly. These frequencies must now be attenuated by the loop—in other words, by the phase transfer function  $H(f)$ , which is a lowpass filter having bandwidth  $B_L$  as defined by [Eq. \(4.9\)](#). To get sufficient noise rejection at higher frequencies, the loop bandwidth should be chosen so as to coincide with the stopband-edge frequency of the highpass filter  $\text{NTF}(f)$ . Moreover, the gain of  $H(f)$  should rolloff sharply above  $B_L$  so we should choose a higher-order loop filter. This will be discussed in greater detail in [Chap. 9](#).

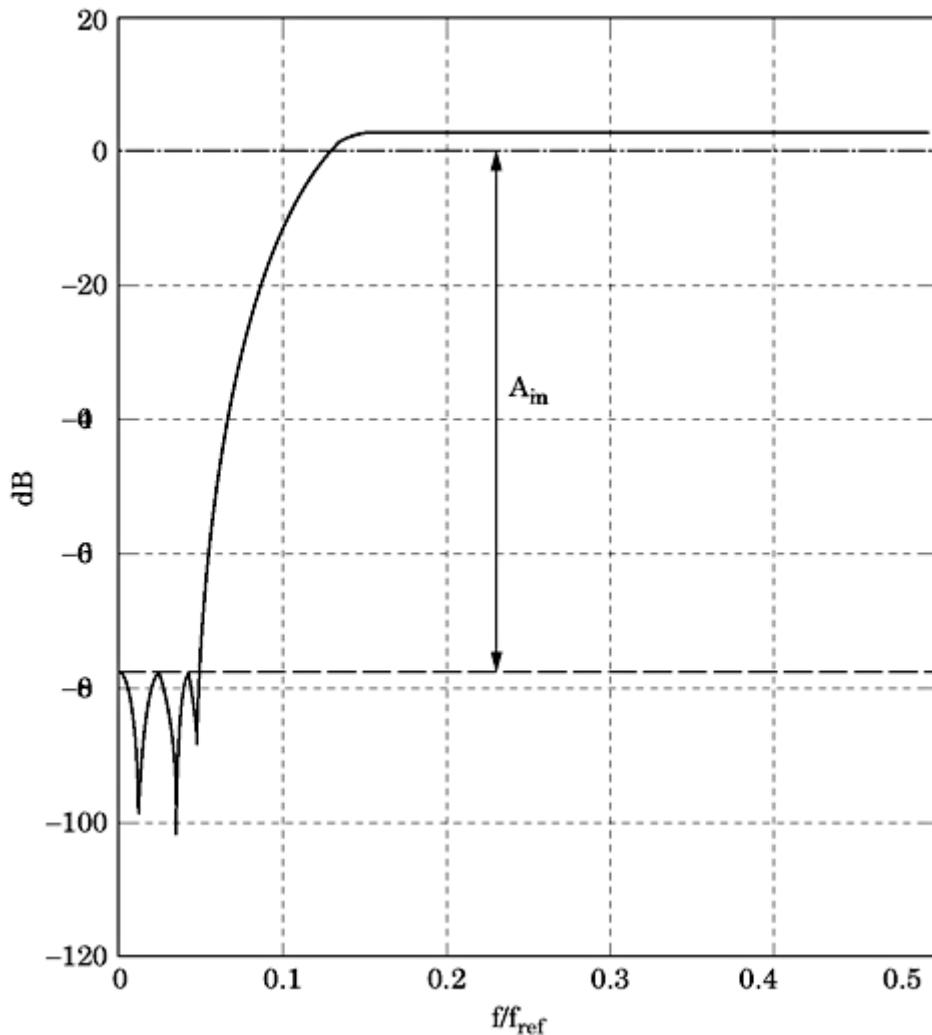
Well, all these formulas are quite abstract, so we may ask what exactly their physical meaning is. In [Sec. 6.7.1](#), it has been shown that the quantity  $S_{\theta\theta,\text{out}}(f)$  can be interpreted as the ratio of noise power  $P_{n,1\text{Hz}}$  within a bandwidth  $B$  of 1 Hz to carrier power  $P_c$ , as indicated in [Eq. \(7.51\)](#)

$$\frac{P_{n,1\text{Hz}}}{P_c} = S_{\theta\theta,\text{out}}(f) \cdot B, \quad B = 1 \text{ Hz} \quad (7.51)$$

In the configuration of [Fig. 7.22](#),  $P_c$  is simply the power of the VCO output signal; hence, knowing the PSD of the phase noise  $S_{\theta\theta,\text{out}}(f)$ , we can estimate how much noise will be present in a 1-Hz bandwidth at any desired frequency  $f$  (which is an offset from the carrier frequency  $f_0$ ). Since the quantity  $S_{\theta\theta,\text{out}}(f)$  is a very small number, it is more practical to use logarithmic units (dB):

$$S_{\theta\theta,\text{out dB}}(f) = 10 \log_{10}(S_{\theta\theta,\text{out}}(f) \cdot B) [\text{dBc/Hz}] \quad (7.52)$$

As we will see in Sec. 7.5, Eqs. (7.50) through (7.52) will be used to determine the required order  $n$  of a  $\Sigma\Delta$  modulator when the maximum allowed ratio of noise power  $P_{n,1\text{Hz}}$  to carrier power  $P_c$  is specified in advance at some frequency  $f$ .



**Figure 7.23** The frequency response of the Chebyshev 2 highpass filter.  $A_{\min}$  is the minimum damping in the stopband.

[Equation \(7.49\)](#) has been derived for the case where the NTF is implemented by a Butterworth highpass filter. When the Chebyshev 2 filter is used, however, the frequency response of the Chebyshev 2 filter must be inserted into [Eq. 7.47](#). As shown in [Fig. 7.23](#), the Chebyshev 2 highpass has constant ripple in the stop-band. When designing such a filter, the minimum damping  $A_{\min}$  is specified first. Given  $A_{\min}$  an upper bound for the noise perturbation  $S_{\theta\theta,\text{out dB}}(f)$  in the stop-band can be estimated from

$$S_{\theta\theta,\text{out dB}}(f) < \left( 10 \log_{10} \left( \frac{2f_{\text{ref}}^2}{f^2} \cdot \frac{1}{12f_{\text{ref}}} \right) - A_{\min} \right) B \quad [\text{dBc/Hz}] \quad (7.53)$$

with  $B = 1$  Hz. When the designer of a  $\Sigma\Delta$  modulator postulates a minimum value for  $S_{\theta\theta,\text{out dB}}(f)$  at some frequency  $f$ , [Eq. \(7.53\)](#) indicates the required minimum damping  $A_{\min}$  in order to

reach that goal.

Before dealing with the details of the  $\Sigma\Delta$  modulator layout, we first put the emphasis on an inconvenient property of  $\Sigma\Delta$  modulators that is neglected by most authors reporting on this topic: the instability created by saturation effects in the quantizer. We will deal with this effect in [Sec. 7.4.4](#).

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).

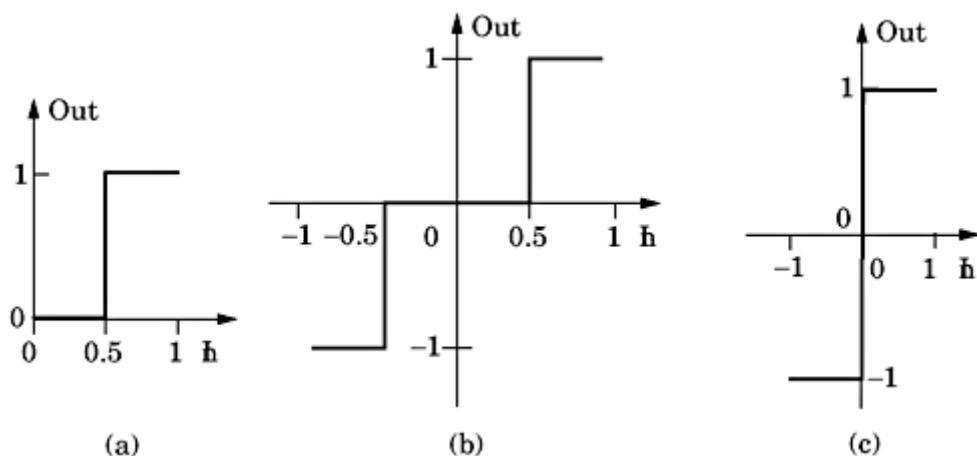
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.

Any use is subject to the Terms of Use as given at the website.

## Nonlinear effects in $\Sigma\omega$ modulators

In the  $\Sigma\Delta$  modulators considered hitherto, a quantizer has been used that has only two quantization levels (0 and 1) with a switching threshold set at an input level of 0.5 (cf. Fig. 7.5b). When realizing a  $\Sigma\Delta$  modulator like the one shown in Fig. 7.21, we would expect the system to operate within a range of 0 to 1. Simulations performed with this configuration revealed, however, that the system is not stable within the whole range but becomes unstable for inputs below approximately 0.2 and above approximately 0.8. When watching the input of the quantizer (for example, by modeling the modulator with Simulink<sup>25</sup> and connecting a “scope” to the quantizer input), we observe that for an input signal of 0.8, the signal sometimes exceeds 1.5 where the quantizer theoretically should switch to output level 2. Such excursions stem from the fact that the noise gain within the system can be greater than 1; hence, higher signal levels are possible. Because the simple two-level quantizer is not able to output level 2, the quantizer output becomes “saturated,” and the loop is not able to feed back the required signal to the input summation block. As a result, the system becomes unstable—that is, the quantization error runs away toward infinity.

$\Sigma\Delta$  modulators built with such simple two-level (cf. Fig. 7.24a) quantizers are therefore not able to work within the required fractional range of 0 to 1. The required range can be obtained using several methods. The most straightforward way would be to set the quantization level to  $Q = 2$  instead of  $Q = 1$ , as depicted in Fig. 7.24c. With  $Q = 2$ , the  $\Sigma\Delta$  modulator would provide stable operation within a range of about  $-0.75$  to  $0.75$ . Doubling  $Q$ , however, also doubles the amplitude of the error sequence  $e(nT)$ , and consequently the power spectral density of the phase perturbation  $S_{\theta\theta,\text{out}}(f)$  is increased by a factor of 4 or 6 dB [cf. Eqs. (7.50) and (7.52)]. When using a quantizer with  $Q = 2$ , the divider ratio of the down scaler must no longer be switched between the values  $N$  and  $N + 1$ , but rather between  $N - 1$  and  $N + 1$ . A more elegant method of increasing the range of fractional numbers would be to implement a three-level quantizer as shown in Fig. 7.24b. The three-level quantizer has switching thresholds



**Figure 7.24** The input-output characteristic of different quantizers. (a) A two-level quantizer with  $Q = 1$ . (b) A three-level quantizer with  $Q = 1$ . (c) A two-level quantizer with  $Q = 2$ .

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

at  $-0.5$  and  $0.5$ , and its output levels are  $-1$ ,  $0$ , and  $1$ , respectively. The quantization step remains at  $Q = 1$ , hence there is no increase in phase perturbation. The output of this quantizer can take the values  $-1$ ,  $0$ , and  $1$ , and the divider ratio of the down scaler must be switched between the ratios  $N - 1$ ,  $N$ , and  $N + 1$ . Simulink simulations with the three-level quantizer reveal that this system is stable within a range from about  $-0.75$  to  $0.75$ . The range  $-0.5$  to  $0.5$  is sufficient to realize any desired fractional division ratio in a fractional- $N$  frequency synthesizer. The problem of limited fractional range in two-level quantizers is largely neglected in the literature on fractional- $N$  frequency synthesizers. One of the few papers dealing with the limited range of two-level quantizers has been written by Shu.<sup>60</sup>

## The Design Procedure for $\Sigma\Delta$ Modulators

Having analyzed the dynamic performance of  $\Sigma\Delta$  modulators in great detail, we are ready now for the practical implementation. The design of a  $\Sigma\Delta$  modulator can be performed in a sequence of steps that will be described in the following.

**Step 1. General specifications.** To specify the parameters of the  $\Sigma\Delta$  modulator, the general specifications of the fractional- $N$  frequency synthesizer must be known. The following parameters should be specified first:

- The frequency  $f_0$  at the output of the VCO (or range of frequencies).
- The divider ratio  $N$  (or the range  $N_{\min}$  to  $N_{\max}$ ). When a range is specified, the mean  $N_{\text{mean}} = \sqrt{N_{\min}N_{\max}}$  should be calculated (cf. Sec. 6.5). When  $N$  is constant,  $N_{\text{mean}} = N$ .
- For  $N_{\text{mean}} = N$ , the natural frequency  $\omega_n$  and damping factor  $\zeta$  have to be determined.
- From  $\omega_n$  and  $\zeta$ , the loop bandwidth  $B_L$  must be computed [cf. Eq. (4.9)].
- The reference frequency  $f_{\text{ref}}$  (frequency at the input of the phase detector).

**Step 2. Oversampling ratio (OSR).** From  $B_L$  and  $f_{\text{ref}}$  the oversampling ratio can be determined:  $\text{OSR} = \frac{f_{\text{ref}}/2}{B_L}$ .

**Step 3. Stopband-edge frequency  $f_c$  of the highpass filter.** The highpass filter NTF( $f$ ) must suppress the low frequencies in the range of  $0$  to  $f_c$ .  $f_c$  should coincide with the loop bandwidth  $B_L$ ; hence, we have  $f_c = \frac{f_{\text{ref}}/2}{\text{OSR}}$ .

**Step 4. Frequency response NTF( $f$ ) of the highpass filter.** The highpass filter can either be realized by a Butterworth or by a Chebyshev 2 function.

**Case study 1: Butterworth highpass filter.** The required order of the  $\Sigma\Delta$  modulator can be determined from Eq. (7.50). The designer will specify a desired power spectral density  $S_{\theta\theta,\text{out}}(f)$  at some frequency  $f$ —for example, the maximum

**Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.**

value  $S_{\theta\theta,\text{out}}(f)$  at  $f = 10$  kHz should not exceed  $-140$  dB/Hz. The required order is then obtained by solving [Eq. \(7.50\)](#) for  $n$ . Order  $n$  and the stopband-edge frequency  $f_c$  of the highpass filter are now known. To complete the design, we also must know the gain of the filter at the Nyquist frequency—in other words,  $\text{NTF}(f_{\text{ref}}/2)$ . First, the maximum noise power gain  $\text{NPG}_{\max}$  must be determined from [Eq. \(7.20\)](#). When a three-level quantizer is used (cf. [Fig. 7.24b](#)), the  $\Sigma\Delta$  modulator should be stable at least within the input range  $-0.5$  to  $0.5$ ; hence, we will specify a value for  $U_A$  that is somewhat larger than  $0.5$ —for instance,  $0.7$ . Given  $n$  and  $U_A$ , the NPG can now be calculated.  $\text{NTF}(f_{\text{ref}}/2)$  is then computed from [Eq. \(7.19\)](#). We are ready now to design the highpass filter. Basically any digital filter design program can be used for this purpose, but unfortunately these programs always realize highpass filters whose gain is  $1$  at the Nyquist frequency. In our application, however, this gain should be greater than  $1$ , usually in the region between  $1.2$  and  $1.5$ . A very powerful design tool has been created by Schreier:<sup>61</sup> the DELSIG toolbox (Delta-Sigma), which is a toolbox to be used with Matlab.<sup>25</sup> The DELSIG toolbox can be downloaded for free from the Web. The Matlab function *synthesizeNTF.m* is used to design the highpass filter. The function is called with a number of input arguments, such as:

- order  $n$
- OSR
- opt: This parameter specifies whether a Butterworth or Chebyshev 2 filter must be implemented.
- H\_inf: This is the gain of the filter at the Nyquist frequency  $\text{NTF}(f_{\text{ref}}/2)$ .

The function returns the poles and zeroes of the desired highpass filter.

**Case study 2: Chebyshev 2 highpass filter.** [Equation \(7.53\)](#) can be used to calculate the minimum damping  $A_{\min}$  in the stopband of the highpass filter when a value for  $S_{\theta\theta,\text{out}}(f)$  at some frequency  $f$  has been specified. Note that specifying  $A_{\min}$  and  $f_c$  (stopband-edge frequency) determines the minimum damping within the stopband for any order  $n$  of the highpass filter; hence, theoretically any filter order  $n$  (even  $n = 1$ ) could be used. Too small a filter order, however, results in a wide transition region between the stopband-edge frequency and passband-cutoff frequency. Matlab's signal processing toolbox provides a function *Cheb2ord* that returns to the required order of a Chebyshev 2 filter when both the stopband-edge and passband-cutoff frequencies are known.

To give a numerical example, assume that  $f_{\text{ref}} = 1$  MHz and OSR = 32. Then, the stopband-edge frequency  $f_c = (f_{\text{ref}}/2)/32 = 15.6$  kHz. It is thus reasonable to postulate that the passband-cutoff frequency  $f_P$  is 50 kHz (it must be greater than  $f_c$  and less than  $f_{\text{ref}}/2$ ). When we require a minimum damping of 80 dB in the stopband and a damping of 3 dB at the passband cutoff frequency, the *Cheb2order* function returns a minimum order of 6.

The maximum filter gain  $\text{NTF}(f_{\text{ref}}/2)$  is determined the same way as the Butterworth filter. The *synthesizeNTF* function of the DELSIG toolbox can be used again to design the required highpass filter  $\text{NTF}(f)$ .

**Step 5. IIR filters  $L_0(z)$  and  $L_1(z)$  (cf. Fig. 7.21).** When structure 1 (Fig. 7.21a) is chosen, set  $\text{STF}(z) = 1$  and use Eqs. (7.28a) and (7.28b) to compute the IIR filter functions  $L_0(z)$  and  $L_1(z)$ . When structure 2 (Fig. 7.21b) has been selected, set  $\text{STF}(z) = z^{-1}$  and use Eq. 7.28b to determine  $L_1(z)$  and Eq. (7.30b) to determine  $L_0(z)$ .

## Spurs, Fractional Spurs, and Subfractional Spurs

In fractional- $N$  frequency synthesizers, three types of spurs can be observed: spurs (at the reference frequency), fractional spurs (at a fraction of the reference frequency), and subfractional spurs (at fractions of the fractional spurs). The first category is created by the phase detector. As discussed in Sec. 6.7.3, the EXOR and JK-flipflop phase detectors give rise to spurious sidebands that are displaced from the VCO output frequency by the reference frequency and multiples thereof. Moreover, when the PFD with voltage output is used, spurs at subharmonics of  $f_{\text{ref}}$  can occur. In modern frequency synthesizers, the PFD with charge pump is the preferred type of phase detector. As we have seen, this type of PFD only exhibits reference frequency spurs when there is a mismatch of the current sources in the phase detector (cf. Fig. 2.16).

As demonstrated in Sec. 7.4.1, the  $\Sigma\Delta$  modulator creates fractional spurs—that is, spurs located at a fraction of the reference frequency and multiples thereof. When the fractional portion  $F$  of the division ratio of the down scaler is written as the ratio of two integers  $F = S/T$ , a first-order  $\Sigma\Delta$  modulator will show up a spur at  $f_{\text{ref}}/T$  and multiples thereof. It is generally agreed that the error sequence  $e(nT)$  of the quantizer becomes more randomized when higher-order converters are implemented. In the literature on  $\Sigma\Delta$  modulators, contradictory statements on fractional spurs are found, however. Ning He and his coauthors<sup>62</sup> present a study where a complex mathematical analysis comes to the conclusion that the error sequence of a second-order  $\Sigma\Delta$  modulator with weighted feedback is fully randomized for a DC input signal. A modulator without dithering was used in this study. The author simulated He's configuration with Simulink and found powerful spurs in the spectrum of the modulator's output signal. The authors' experience is in full agreement with the statements of Norsworthy.<sup>55</sup> Here, also,  $\Sigma\Delta$  modulators of various orders were implemented with and without dithering, and it became very clear that even higher-order converters (order 4 and more) still are plagued by spurs when no dithering is applied. Manufacturers of commercial fractional- $N$  frequency synthesizers also provide dithering—for example, in the LMX2470 of National Semiconductors.<sup>54</sup> This circuit can be configured to work with or without dithering. The author therefore comes to the conclusion that fractional spurs are found in  $\Sigma\Delta$  modulators of any order when no dithering is applied, but that the spurs vanish almost completely with dither. A third category of spurs is subfractional spurs.

**Any use is subject to the Terms of Use as given at the website.**

In third- and higher-order  $\Sigma\Delta$  modulators, additional spurs are created at half the frequencies of fractional spurs.

## Alternative $\Sigma\Delta$ Modulators: The MASH Converter

Higher-order  $\Sigma\Delta$  modulators can be realized by another architecture, called MASH (multi-stage noise shaping). With the MASH technique, higher-order  $\Sigma\Delta$  modulators are implemented by cascading lower-order modulators. A fifth-order  $\Sigma\Delta$  modulator, for example, can be obtained by cascading a second-order and a third-order modulator, or by cascading one first-order and two second-order modulators, or by cascading five first-order modulators. In this section, we will concentrate on cascading first-order modulators.

The building block to be used in MASH converters is shown in Fig. 7.25a. It is identical with the already discussed first-order  $\Sigma\Delta$  modulator but has an additional output  $-E$ , which represents the error sequence. The error signal is created by an additional summing block that computes the error sequence from the difference  $V - Y$ . (The reason for the minus sign will be explained in the following.) To simplify block diagrams of the MASH converters to be presented next, the first-order  $\Sigma\Delta$  modulator in Fig. 7.25a will be represented by the shortcut in Fig. 7.25b. Applying the theory discussed in Sec. 7.4.1, the output  $Y$  of the first-order converter in Fig. 7.25a is

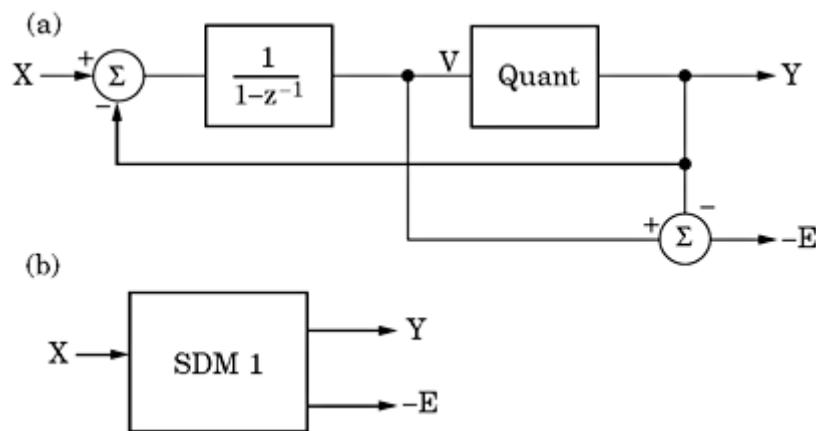
$$Y(z) = X(z)z^{-1} + (1 - z^{-1})E(z) \quad (7.53)$$

thus, the signal transfer function STF( $z$ ) of this modulator is

$$\text{STF}(z) = z^{-1}$$

and the noise transfer function is

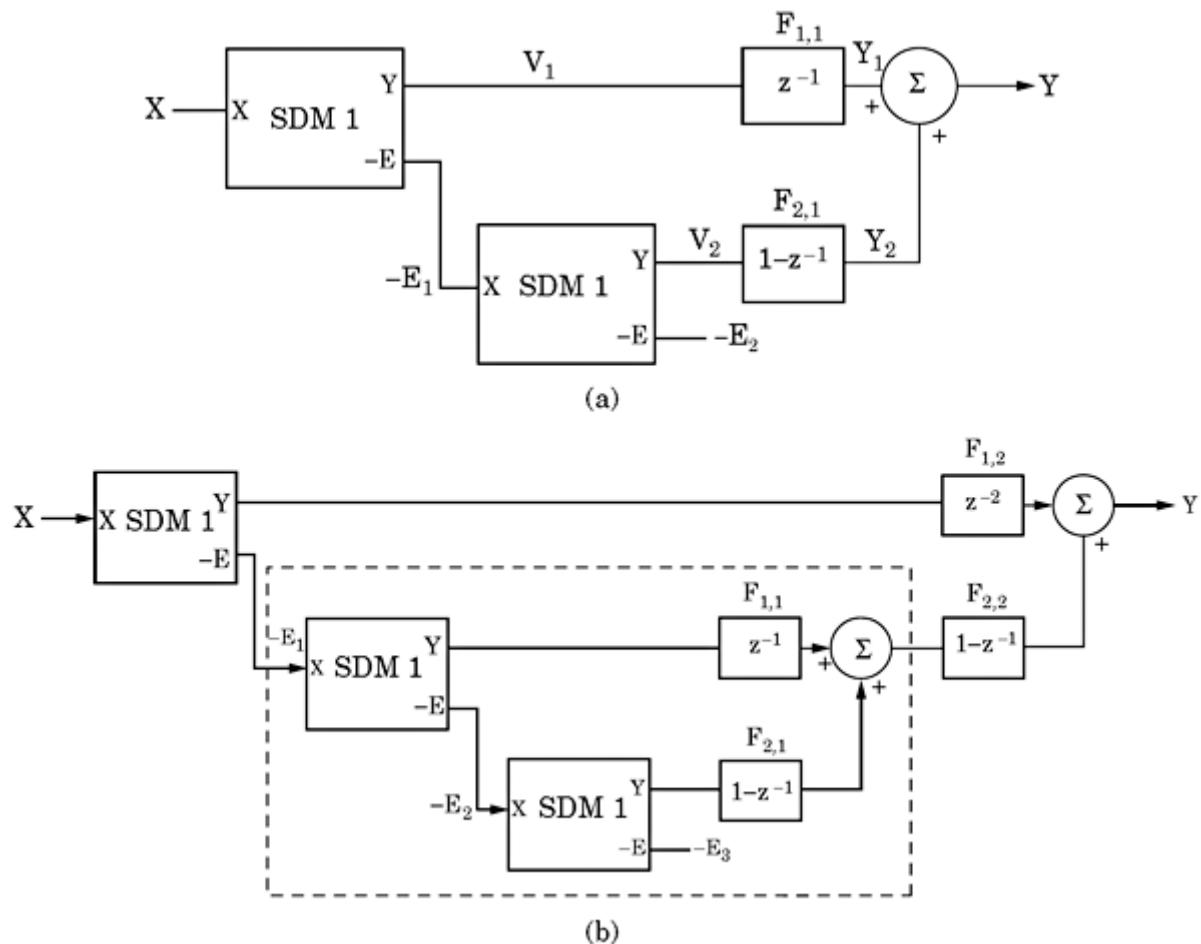
$$\text{NTF}(z) = 1 - z^{-1}$$



**Figure 7.25** A block diagram of first-order MASH converter. (a) A detailed block diagram. (b) A shortcut for the first-order MASH converter.

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.



**Figure 7.26** Higher-order MASH converters. (a) A second-order MASH converter built from two first-order blocks. (b) A third-order MASH converter built cascaded from three first-order blocks.

which corresponds to a simple differentiator. A second-order MASH converter is shown in Fig. 7.26a. It consists of a cascade of two first-order modulators and two additional filter blocks  $F_{1,1}$  and  $F_{2,1}$ . The motivation for these filter blocks will become evident in the following analysis.

The output signal  $V_1(z)$  of the first modulator block is

$$V_1(z) = X(z)z^{-1} + E_1(z)(1 - z^{-1})$$

The output signal  $V_2(z)$  of the second modulator is

$$V_2(z) = -E_1(z)z^{-1} + E_2(z)(1 - z^{-1})$$

For the signals  $Y_1(z)$  and  $Y_2(z)$  at the outputs of the filters, we get

$$Y_1(z) = X(z)z^{-1}F_{1,1}(z) + E_1(z)(1 - z^{-1})F_{1,1}(z)$$

$$Y_2(z) = -E_1(z)z^{-1}F_{2,1}(z) + E_2(z)(1 - z^{-1})F_{2,1}(z)$$

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

and the final output  $Y(z)$  becomes

$$\begin{aligned} Y(z) &= Y_1(z) + Y_2(z) = X(z)z^{-1}F_{1,1}(z) \\ &\quad + E_1(z)[(1 - z^{-1})F_{1,1}(z) - z^{-1}F_{2,1}(z)] \\ &\quad + E_2(z)[(1 - z^{-1})F_{2,1}(z)] \end{aligned} \quad (7.54)$$

The output signal therefore contains two noise terms from both error sequences  $e_1(t)$  and  $e_2(t)$ . We can cancel the contribution by  $e_1(t)$  by making the coefficient of the  $E_1$  term in Eq. (7.54) zero. This yields the equation

$$(1 - z^{-1})F_{1,1}(z) - z^{-1}F_{2,1}(z) = 0$$

which leads to the relation

$$\frac{F_{1,1}(z)}{F_{2,1}(z)} = \frac{z^{-1}}{1 - z^{-1}}$$

The contribution of  $e_1(t)$  is canceled when we set

$$\begin{aligned} F_{1,1}(z) &= z^{-1} \\ F_{2,1}(z) &= 1 - z^{-1} \end{aligned} \quad (7.55)$$

Under this condition, the final output signal  $Y(z)$  becomes

$$Y(z) = X(z)z^{-2} + E_2(z)(1 - z^{-1})^2 \quad (7.56)$$

Hence, the signal transfer function of the second-order MASH converter becomes

$$\text{STF}_2(z) = z^{-2}$$

which corresponds to a delay by two sampling intervals, and the noise transfer function becomes

$$\text{NTF}_2(z) = (1 - z^{-1})^2 \quad (7.57)$$

which corresponds to double differentiation.

To increase the order of this MASH converter by 1, we simply have to cascade the configuration in Fig. 7.26a with another first-order  $\Sigma\Delta$  modulator. The resulting system is plotted in Fig. 7.26b. The block enclosed in the dashed rectangle is the second-order

modulator. To eliminate the contribution of the error sequence  $e_1(t)$  to the final output signal  $y(t)$ , two blocks labeled  $F_{1,2}$  and  $F_{2,2}$  are added. An analogous computation reveals that for these two blocks the transfer functions must be chosen

$$\begin{aligned}F_{1,2}(z) &= z^{-2} \\F_{2,2}(z) &= 1 - z^{-1}\end{aligned}\tag{7.58}$$

Under this condition, we get for the final output signal

$$Y(z) = X(z)z^{-3} + E_3(z)(1 - z^{-1})^3$$

thus, the signal transfer function gets  $\text{STF}_3(z) = z^{-3}$ , which corresponds to a delay by three sampling intervals, and the noise transfer function becomes  $\text{NTF}_3(z) = (1 - z^{-1})^3$ , which signifies triple differentiation. To further increase the order of the MASH converter, an identical scheme is used. To get a fourth-order MASH converter, the lower filter block  $F_{2,3}$  would still be a simple differentiator—that is,  $F_{2,3}(z) = 1 - z^{-1}$ , whereas for the upper filter block we would have to use  $F_{1,3}(z) = z^{-3}$ , and so on. For each additional stage, the order of the filter  $F_{1,n}$  is increased by 1, but the filter  $F_{2,n}$  stays the same. For arbitrary modulator order  $n$ , the final output signal  $y(t)$  is given by

$$Y(z) = X(z)z^{-n} + E_n(z)(1 - z^{-1})^n \quad (7.59)$$

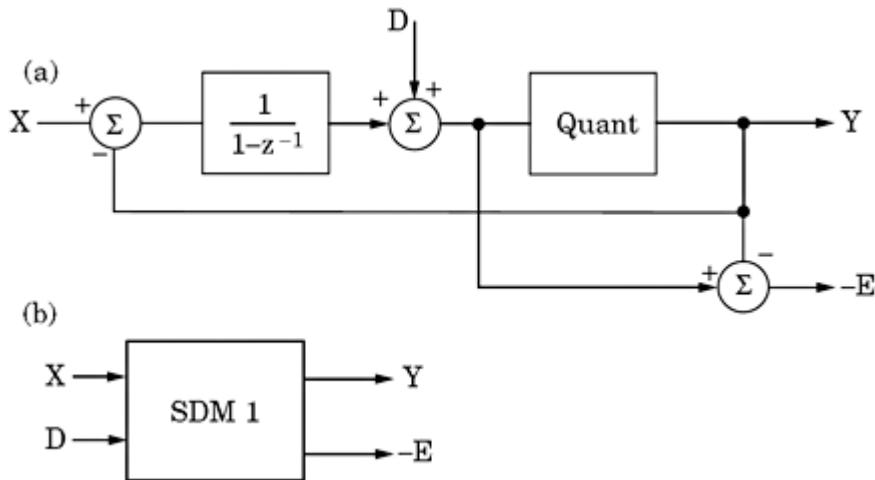
which says that the error sequence is differentiated  $n$  times.

In the MASH converters discussed hitherto, every feedback loop includes one integrator at most; hence, the systems are always stable. The output signal  $y(t)$  of MASH converters, however, differs significantly from that of the previously analyzed higher-order  $\Sigma\Delta$  modulators as, for example, in Fig. 7.21. Looking at the second-order system in Fig. 7.26a reveals that the final output signal is the sum of two local output signals—in other words, a quantizer output signal and a differentiated quantizer output signal. Assuming that a two-level quantizer is used throughout the output of the first quantizer can have the values 0 or 1, whereas the differentiated output of the second quantizer can take values  $-1, 0$ , or  $1$ . The sum of the signals therefore can be in the range  $-1$  to  $2$ —that is, they can take four different levels. When this MASH converter is used to control the division ratio of a down scaler, this ratio is not only switched between two levels  $N$  and  $N + 1$ , respectively, but between four levels, say  $N, N + 1, N + 2$ , and  $N + 3$ . It is easily seen that for the third-order MASH converter (cf. Fig. 7.26b), the final output signal can even take eight different levels, which are in the range from  $-3$  to  $4$ . For an arbitrary order  $n$ , the number of output levels becomes  $2^n$ .

Like any other  $\Sigma\Delta$  modulator, the MASH converters also are subject to fractional and subfractional spurs. To reject such spurs, dithering is generally applied. Fig. 7.27a shows the block diagram of a first-order modulator that is expanded for insertion of a dither signal  $D$ . The dither is added immediately to the input signal of the quantizer. It is a uniformly distributed random signal in the range of  $-Q/2$  to  $Q/2$  with  $Q$  = quantization step (in our example,  $Q = 1$ ). To simplify the following block diagrams, the shortcut of Fig. 7.27b will be used for the dithered first-order modulator.

Next, we are going to realize a dithered second-order MASH converter (cf. Fig. 7.28). Each stage of this system must have its own independent dither source<sup>55</sup>—that is, there must be no correlation between the dither sources  $d_1$  and  $d_2$ . Normally, pseudorandom sequences are used. As will be seen, the

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.



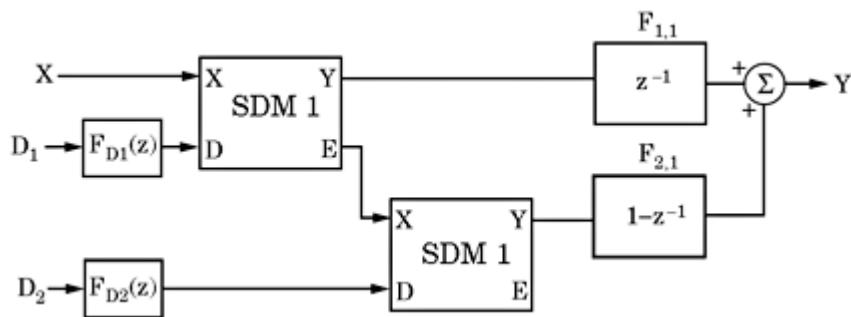
**Figure 7.27** A first-order modulator with additional dither input. (a) A block diagram of the first-order modulator. (b) A shortcut for the first-order modulator with dither input.

dither sources must be prefiltered (except the one in the final stage). To determine the required type of prefilter(s), the final output  $y(t)$  is computed as a function of all input signals  $x(t)$ ,  $e_2(t)$ ,  $d_1(t)$ , and  $d_2(t)$ .

The  $z$ -transform  $Y(z)$  of the final output signal  $y(t)$  is

$$\begin{aligned}
 Y(z) = & X(z) \\
 & + E_2(z)(1 - z^{-2}) \\
 & + D_1(z)[z^{-1}F_{D1}(z)(1 - z^{-1})] \\
 & + D_2(z)[(1 - z^{-1})^2F_{D2}(Z)]
 \end{aligned} \tag{7.60}$$

The noise performance of the MASH converter becomes optimum when the transfer functions of the dithers, as seen in the final output, are the same as the transfer function of the error sequence  $e_2(t)$ , which is a second-order differentiation term. Because the noise transfer function for  $D_2(z)$  already has such a term, prefiltering is not necessary at the final stage—thus,  $F_{D2}(z)$  can



**Figure 7.28** A second-order MASH converter using dithering.

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

be set to 1. To get second-order noise shaping for dither  $d_1(t)$ , however, we set  $F_{D1}(z) = 1 - z^{-1}$ —that is, we differentiate the dither signal in the first stage. This result can easily be generalized for arbitrary order  $n$ . It shows up that in an  $n$ th-order MASH converter the dither in stage  $n$  is applied without pre-filtering. In stage  $n - 1$ , the dither source is prefiltered by a first-order differentiator  $F_{D,n-1}(z) = 1 - z^{-1}$ ; in stage  $n - 2$ , the dither source is prefiltered by a second-order differentiator  $F_{D,n-2}(z) = (1 - z^{-1})^2$ , and so on.

For an  $n$ th-order MASH converter, the noise seen at the final output corresponds to an error sequence differentiated  $n$  times, as indicated in [Eq. \(7.59\)](#). The corresponding noise transfer function NTF( $z$ ) is therefore

$$|\text{NTF}(f)| = 2^n \sin\left(\frac{\pi f}{f_F}\right)^n$$

[cf. [Eq. \(7.15\)](#)]. This corresponds to a Butterworth highpass filter, and for the PSD of the phase perturbation at the output of the VCO, Eqs. [\(7.49\)](#) and [\(7.50\)](#) can be used.

Last but not least, it should be noted that in MASH converters using two-level quantizers having  $Q = 1$ , the range of fractional numbers is limited below the required range of  $-0.5$  to  $0.5$ . The situation is the same as in the case of the conventional  $n$ th-order  $\Sigma\Delta$  modulator. The problem can be fixed using the methods discussed previously—in other words, either by increasing the quantization level to  $Q = 2$  or by using three-level quantizers.

## Mixed-Signal PLL Applications Part 3: Miscellaneous Applications

Frequency synthesis is one of the most important PLL applications and was the subject of Chaps. 6 and 7. In this chapter, we will deal with some other typical PLL applications, such as clock signal recovery (cf. Sec. 8.1). Moreover, the PLL has proven extremely useful in the field of motor-speed control, which will be covered in Sec. 8.2.

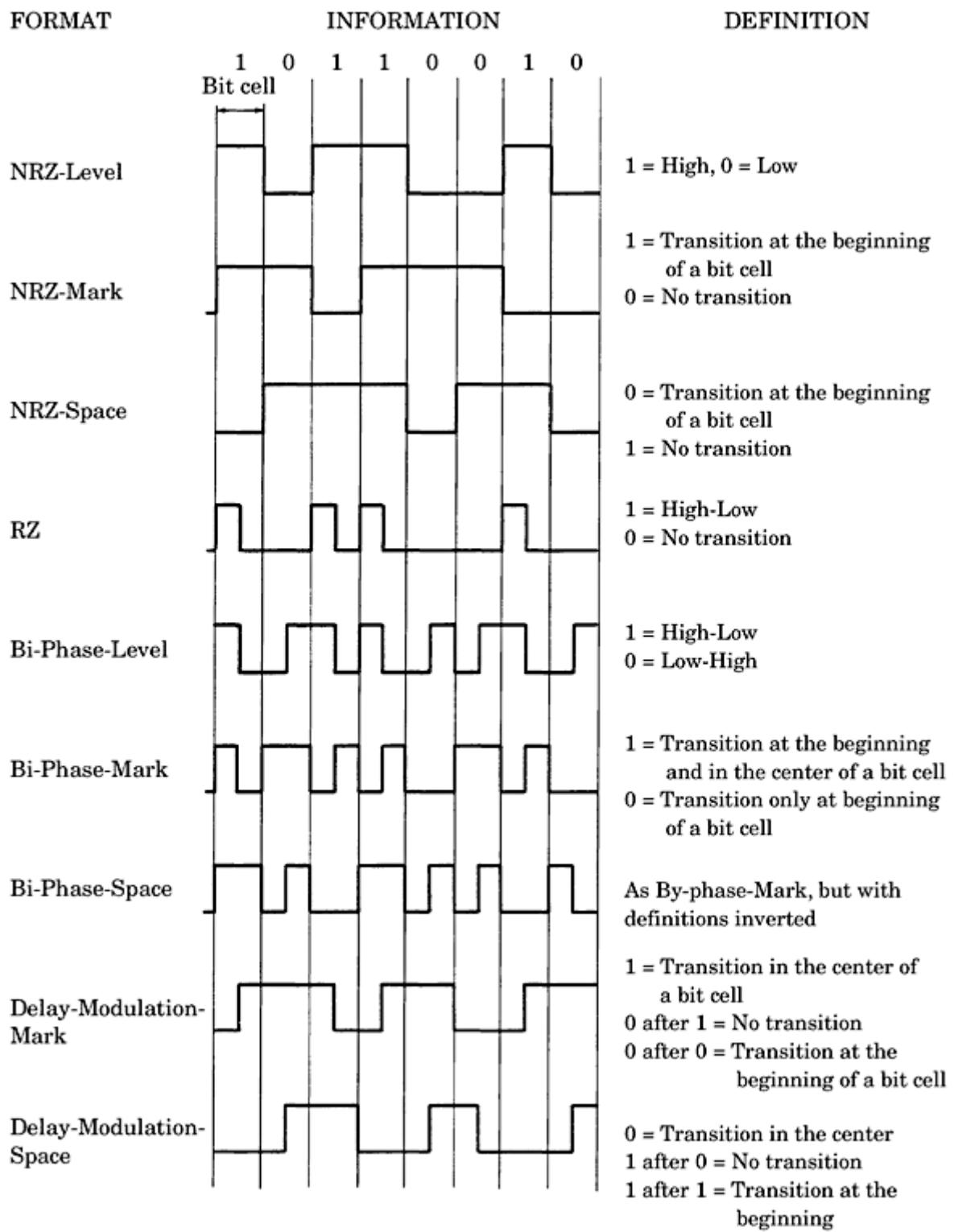
### Retiming and Clock Signal Recovery

The PLL is used in almost every digital communication link. Digital data can be transmitted over serial or parallel data channels. In any case, the data are clocked—that is, they are sent in synchronism with a clock signal, which is normally not transmitted. Thus, the receiver is forced to extract the clock information out of the received signal.

Basically two different principles are used for transmitting digital data: *base-band* and *carrier-based* transmission. In baseband transmission, the digital signal is directly sent over the link; when a carrier system is used, however, the digital signal is first modulated onto a carrier, usually a high-frequency signal. Amplitude (AM), frequency (FM), and phase (PM) modulation are used here, or it is even possible to combine different modulation techniques, such as AM + PM.<sup>20</sup> In carrier systems, a number of digital signals can be modulated onto different carriers; hence, the bandwidth of carrier systems can be much larger than the bandwidth of baseband systems. An overview of these digital modulation schemes is given in Chap. 14.

In this section, we deal only with baseband systems. Let us assume that a digital signal—that is, an arbitrary sequence of 0s and 1s—has to be transmitted over a serial link, using, for example, the familiar RS-232, RS-422, or the RS-485 standard.<sup>28</sup> As we will immediately see, this problem is more serious than would be expected at first glance. First of all, it would seem obvious to transmit the signal such that the 1s are represented as a “high” voltage level and the 0s as

a “low” voltage level, such as 5 V and 0 V, respectively. This simplest data format is sketched in the first row of [Fig. 8.1](#) and is referred to as *NRZ Code* (NRZ = Non Return to Zero). In effect, this is the most commonly used code in *asynchronous* communications. In asynchronous communication, only one single character (an ASCII character in most cases) is transmitted in one message.



**Figure 8.1** Review of the most commonly used binary and pseudo-ternary formats.

Such a character mostly consists of a series of eight data bits. The data bits are headed by a start bit. A stop bit is transmitted after the data bits, and eventually a parity bit is used to enable the receiver to detect single-bit errors. The voltage level on the transmission line is usually “high” when no data are transmitted. The start bit is always a “low” level. The data bits can be “high” or “low,” but the stop bit is always “high.” This ensures that every message starts with a high-to-low transition of the data signal, so the receiver “knows” exactly when the transmission of a character starts.

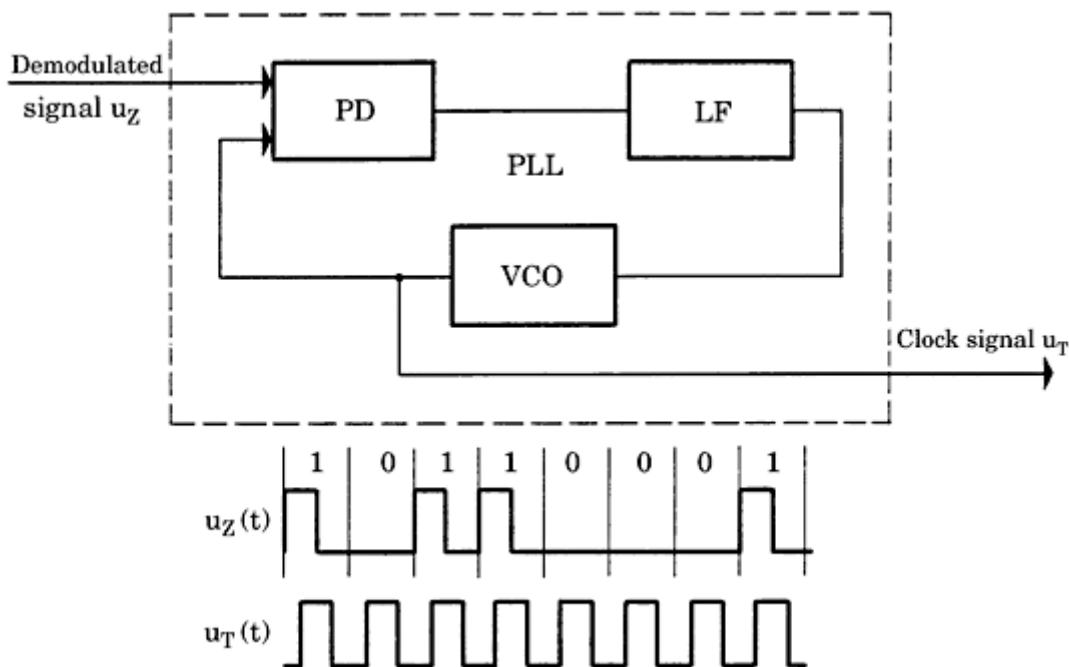
The situation becomes more serious, however, when not a few but hundreds or thousands of bits have to be transmitted in succession. If the NRZ code was used and, say, a thousand 1s were transmitted in succession, no clocking information would be available during that interval. If the clock generator within the data receiver drifts only slightly away, the receiver will not be able to decide whether 999, 1000, or 1001 1s have been sent.

To provide more clocking information, a number of other codes have been devised, as shown in [Fig. 8.1](#). To start with the simplest, the NRZ-mark code produces a level transition whenever a 1 is transmitted. On the other hand, the NRZ-space code generates an edge on the transmission of a 0. As is easily seen, the NRZ-mark code easily fails when a long sequence of 0s is sent; the same problem occurs with the NRZ-space code when a series of 1s are transmitted. The RZ code (return to zero) generates a sequence of “high-low” whenever a 1 is transmitted. This code is prone to failure if long sequences of 0s occur in the data stream. A better choice is the biphase-level code, which produces a sequence “high-low” for a 1 and a sequence “low-high” for a 0. This code contains at least one edge in every bit cell. (A bit cell is the time interval in which one bit is transmitted.) Because clock recovery is very simple with this code, the biphase code is the most often used in high-speed data communications. A drawback of this code is the fact that the required bandwidth to transmit a given number of bits per second is twice the bandwidth of the NRZ code.<sup>20</sup>

There is another code called *delay-modulation code* which combines the advantages of the biphase and NRZ codes. In the delay-modulation-mark code, a transition in the center of a bit cell is generated on the transmission of a 1. If a 0 follows a 1, no transition takes place in the next bit cell. If another 0 follows the first 0, however, a transition at the start of the next bit cell is generated. The bandwidth of this code is almost the same as for the NRZ code.

From this discussion, it’s evident that the methods of extracting clock information must depend on the particular code used. Let’s have a look at the most important recovery techniques.

We shall start with recovering the clock signal for the RZ format ([Fig. 8.1](#)). The corresponding block diagram is shown in [Fig. 8.2](#). Clock signal recovery is accomplished by one PLL. The center frequency of this PLL is chosen to be approximately equal to the baud rate of the data signal. The PLL is synchronized by the transitions of the demodulated data signal, as shown in [Fig. 8.2](#). If an EXOR PD is used, the VCO generates a square-wave signal  $u_T$ , which is in quadrature to the demodulated data signal  $u_z$ , as seen from the waveforms



**Figure 8.2** Operating principle for clock recovery in the RZ format.

in Fig. 8.2. This phase relationship is advantageous, for the positive transitions of the VCO output signal can be used to strobe the data signal.

Synchronization of the clock-recovery PLL takes place on every logic 1 contained in the data signal. During a succession of logic 0s, the VCO continues to oscillate at its instantaneous frequency. Extended sequences of 0s must be avoided since the frequency of the VCO could drift away to the extent that synchronization gets lost.

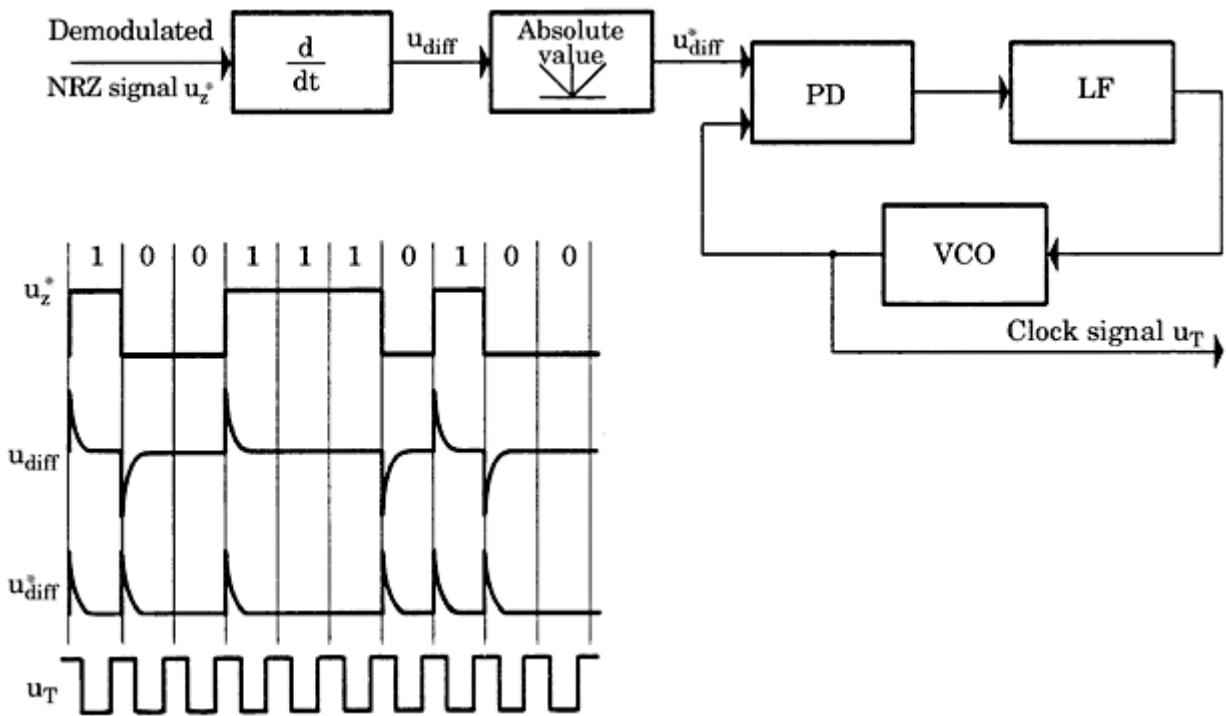
Longer sequences of 0s are avoided if *parity checking* is used. In parity checking, an additional bit of information is added to each group of, say, eight successive data bits. When odd parity is chosen, the total number of 1s, including the parity bit, must be *odd*. The choice of odd parity then ensures that in every sequence of nine bits there is *at least one bit* that is not a 0.

One problem still needs attention: the problem of *initialization*. Every message is finished and a pause will follow the message. During the pause, no signal is transmitted, a situation which is identical to a long sequence of 0s in the case of the RZ code. When a new message is started, synchronization is likely to be lost. Synchronization thus must be re-established. This is done via a fixed preamble that precedes every message. In the case of the RZ code, a typical preamble consists of a series of 1s.

As stated earlier, the major drawback of the RZ format is the large bandwidth required. The NRZ format needs about half that bandwidth, but clock signal recovery is slightly more difficult because the frequency spectrum of the NRZ does not necessarily contain a spectral line at the clock frequency.<sup>1.20</sup>

One method of extracting the clock frequency from the NRZ signal is shown in Fig. 8.3. The demodulated NRZ signal  $u_z^*$  is first differentiated. Then, the differentiated signal  $u_{\text{diff}}$  is rectified by an absolute value circuit.<sup>20.34</sup> As seen in the waveforms in Fig. 8.3, the rectified signal contains a frequency component synchronous with the clock. Hence, it can be used to

synchronize a PLL; the

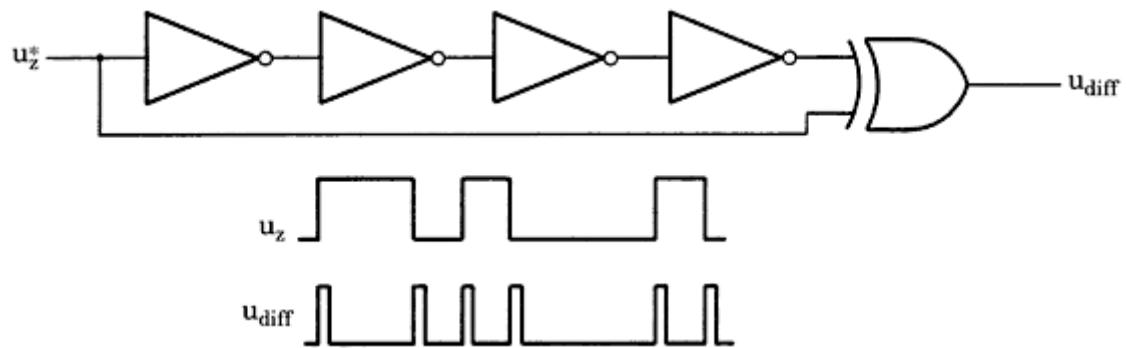


**Figure 8.3** Operating principle for clock recovery in the NRZ format.

output signal of its VCO is the recovered clock signal  $u_T$ . Many analog differentiator and absolute-value circuits have been developed; both operations can be performed alternatively by one single digital circuit.

[Figure 8.4](#) shows a so-called *edge-detector* circuit in which propagation delays of gates are used to produce a pulse on each transition of the input signal. The rise and fall times of the input signal must be shorter than the cumulative propagation delay of the four cascaded inverters. If this condition is not met, a Schmitt trigger must be used to clean up the transition.

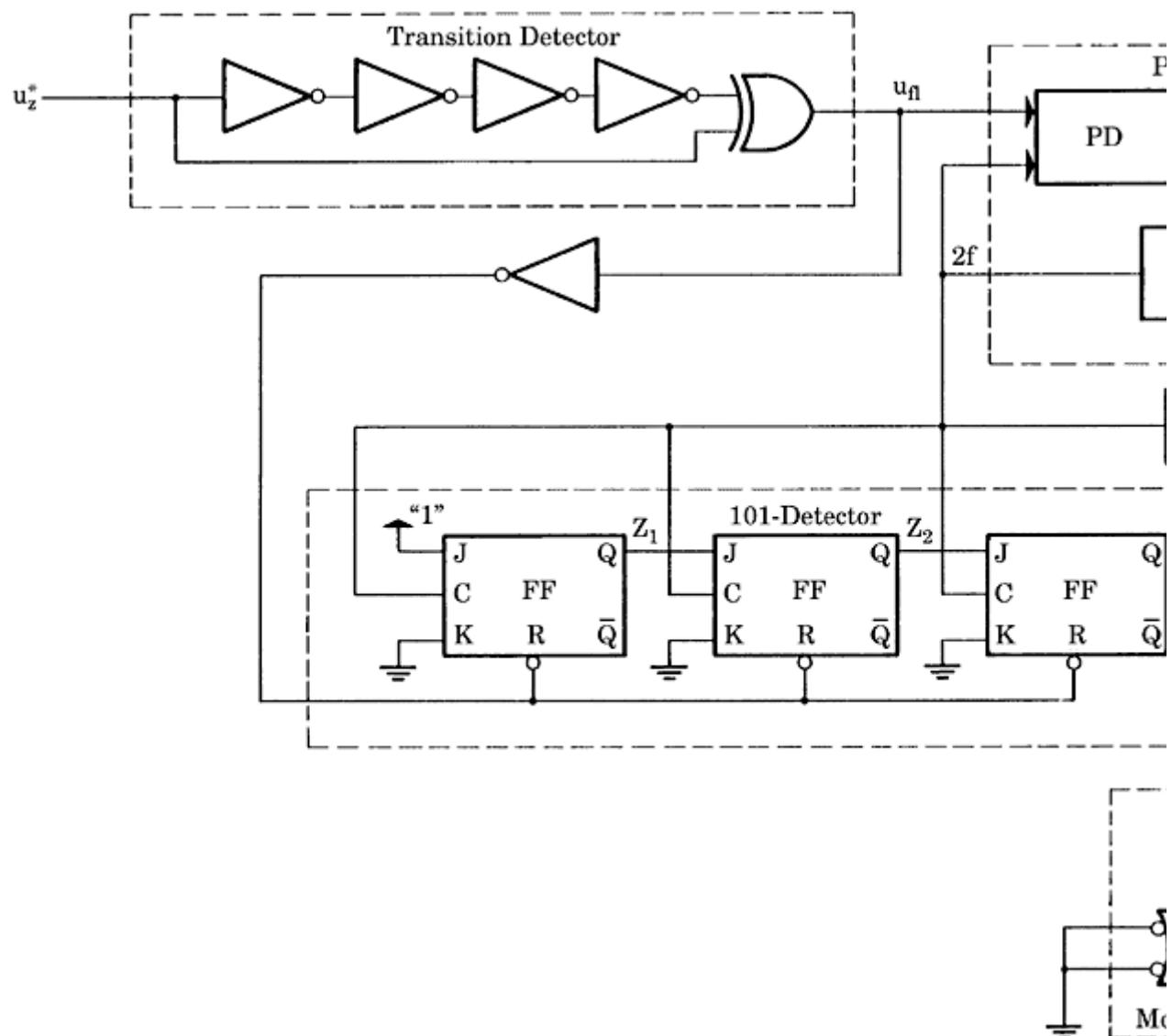
If the biphase or the delay-modulation format is utilized, the demodulated data signal  $u_z^*$  can have transitions at both the beginning and the center of a bit cell, as pointed out in [Fig. 8.1](#). Like in the circuits in Figs. [8.2](#) and [8.3](#), the demodulated signal  $u_z^*$  can be used here to synchronize a PLL operating at twice the clock frequency. A circuit recovering the clock signal for the delay-modulation (DM) format is shown in [Fig. 8.5](#).<sup>27</sup> The waveforms produced by this device are depicted in [Fig. 8.6](#).



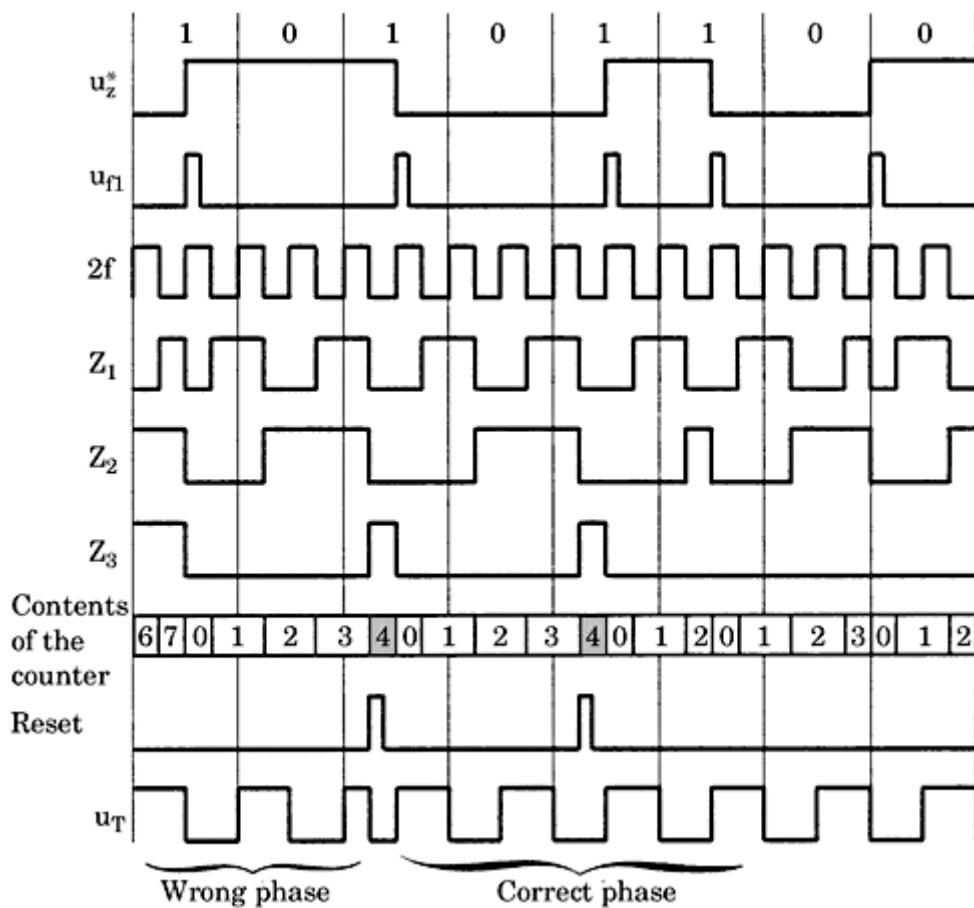
**Figure 8.4** The function of an analog differentiator followed by an absolute-value circuit can be alternatively performed by a digital edge detector.

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
 Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
 Any use is subject to the Terms of Use as given at the website.



**Figure 8.5** A circuit for a clock signal recovery with the DM (delay modulation) format. All flipflop edge of the clock signal  $C$ . Input  $B$  of the monostable multivibrator triggers on the positive on the negative-going edge. Inputs A1 and A2 are unused here.



**Figure 8.6** Waveforms of the circuit of [Fig. 8.5](#).

An edge-detector circuit produces a short positive pulse  $u_{f1}$  on every transition (positive and negative) of the demodulated signal  $u_z^*$ . The signal  $u_{f1}$  is used to synchronize a PLL that operates at twice the clock frequency  $2f$  ([Fig. 8.6](#)). The output signal of the VCO ( $2f$ ) is scaled down in frequency by a factor of 2; the rightmost JK-flipflop of [Fig. 8.5](#) is used for this purpose.

The clock signal  $u_T$  is now defined to be LOW in the first half of every bit cell and HIGH in the second half. We can see from [Fig. 8.5](#) that the circuit could also settle at the *opposite phase* ( $u_T$  being HIGH in the first half of the bit cells). This would result in a faulty interpretation of the received data because the start and the center of every bit cell are erroneously exchanged.

To establish the correct phase of the recovered clock signal, it is necessary initially to reset the JK-flipflop at the right time. For clarity, assume that the recovered clock signal  $u_T$  really has the *wrong phase* at the beginning, as shown in [Fig. 8.6](#). An additional circuit is needed, which takes corrective action. Consider again the signal  $u_{f1}$  in [Fig. 8.6](#). The time interval between any two consecutive pulses of  $u_{f1}$  is not constant but can show the values of 1, 1.5, or 2 periods of a bit cell. An interval longer than two periods is impossible. If a three-stage binary UP counter (labeled as the 101 detector in [Fig. 8.5](#)) is used to count the (negative)

transitions of the signal  $2f$  and if this counter is reset by every  $u_{fl}$  pulse, its content never exceeds 4. Moreover, as we can clearly see in Fig. 8.6, the content of 4 can only be obtained in the first half of a bit cell, and never in the second half.

This fact is used to properly reset the divide-by-2-flipflop in Fig. 8.5. Whenever the three-bit counter reaches 4, a monostable multivibrator (single shot) is triggered. This resets the JK-flipflop. As shown by the waveforms, the phase of the recovered clock is set to its correct state. To enable the corrective action, a preamble of the form 101 ⋯ should precede every message.

It is no major problem to conceive clock-recovering circuits for other formats. As shown in the three examples discussed, the PLL is the key element in each of the applications.

## Motor-Speed Control

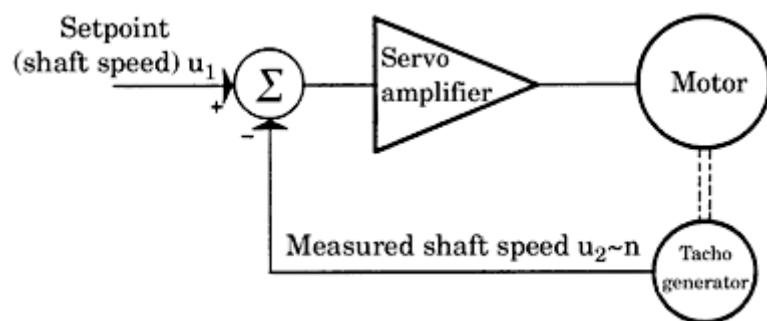
Very precise motor-speed controls at low cost are possible using the PLL. The advantages offered by the PLL technique become evident if the PLL motor-speed controls are first compared with conventional motor-speed controls. A classical scheme for motor-speed control is sketched in Fig. 8.7. The setpoint for the motor speed is given by the signal  $u_1$ . The shaft speed of the motor is measured with a tachometer; its output signal  $u_2$  is proportional to the motor speed  $n$ .

Any deviation of the actual speed from the setpoint is amplified by the servo amplifier whose output stage drives the motor. The gain of the servo amplifier is usually high but finite; to drive the motor, a nonzero error must exist.

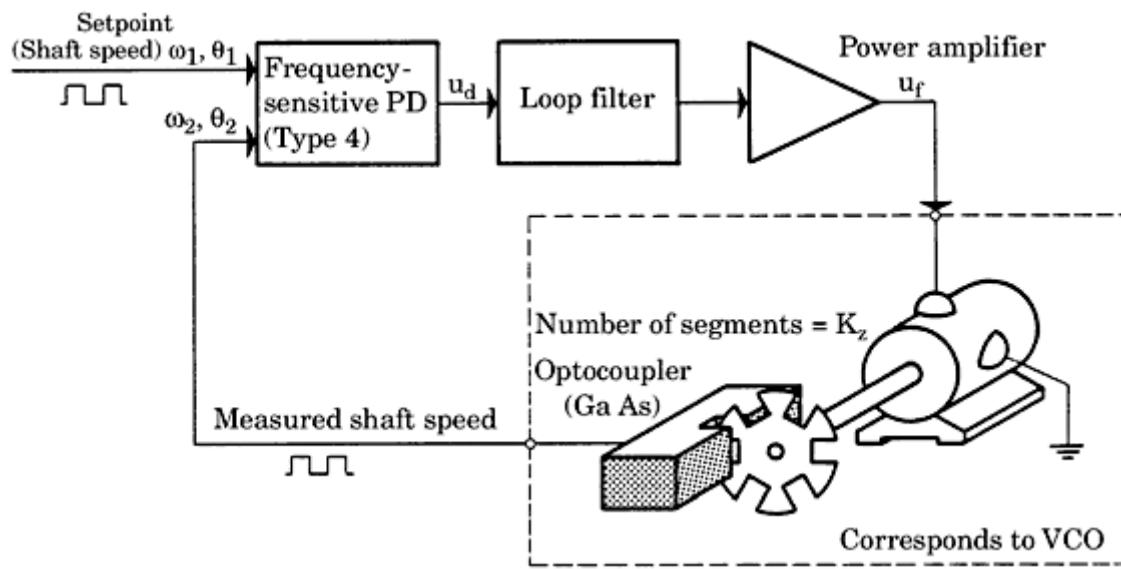
Other sources of errors are nonlinearities of the tachometer and drift of the servo amplifier. A further drawback is the relatively high cost of the tachometer itself.

The PLL technique offers a much more elegant solution. Figure 8.8 is the block diagram of a PLL-based motor-speed control system. The entire control system is just a PLL in which the VCO is replaced by a combination of a motor and optical tachometer, as shown in Fig. 8.8.

The tachometer signal is generated by a fork-shaped optocoupler in which the light beam is chopped by a sector disk (the detailed circuit is shown in Fig. 8.9). The optocoupler is usually fabricated from a light-emitting diode (LED) and a silicon phototransistor. In order to obtain a clean square-wave output, the phototransistor stage is normally followed by a Schmitt trigger, such as a 74HC/HCT14-type CMOS device.



**Figure 8.7** Block diagram of a conventional motor-speed control system.



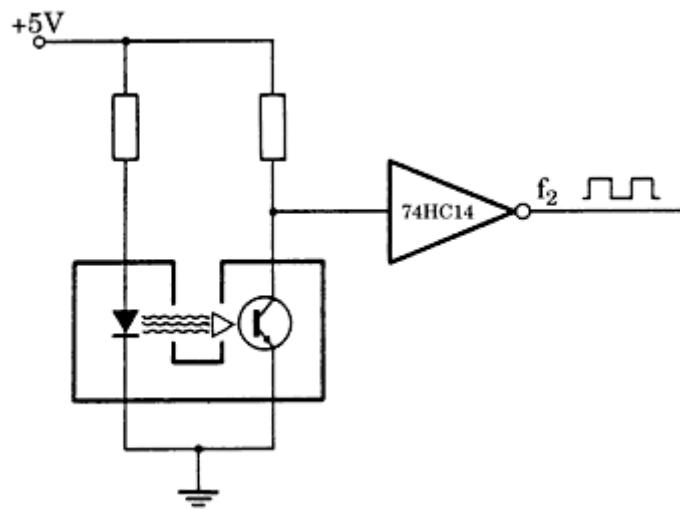
**Figure 8.8** Block diagram of a motor-speed control system based on PLL techniques.

The signal generated by the optocoupler has a frequency proportional to the speed of the motor. Because the phase detector compares not only the radian frequencies  $\omega_1$  and  $\omega_2$  of the reference and the tachometer signals but also *their phases*, the system settles at *zero-velocity error*. To enable locking at every initial condition, the PFD is used as a phase detector.

To analyze the stability of the system, the transfer functions of all blocks in Fig. 8.8 must be known. The transfer functions of the PD and of the loop filter are known, but the transfer function of the motor-tachometer combination still must be determined. If the motor is excited by a voltage step of amplitude  $u_f$ , its angular speed  $\omega(t)$  will be given by

$$\omega(t) = K_m u_f \left[ 1 - \exp\left(-\frac{t}{T_m}\right) \right] \quad (8.1)$$

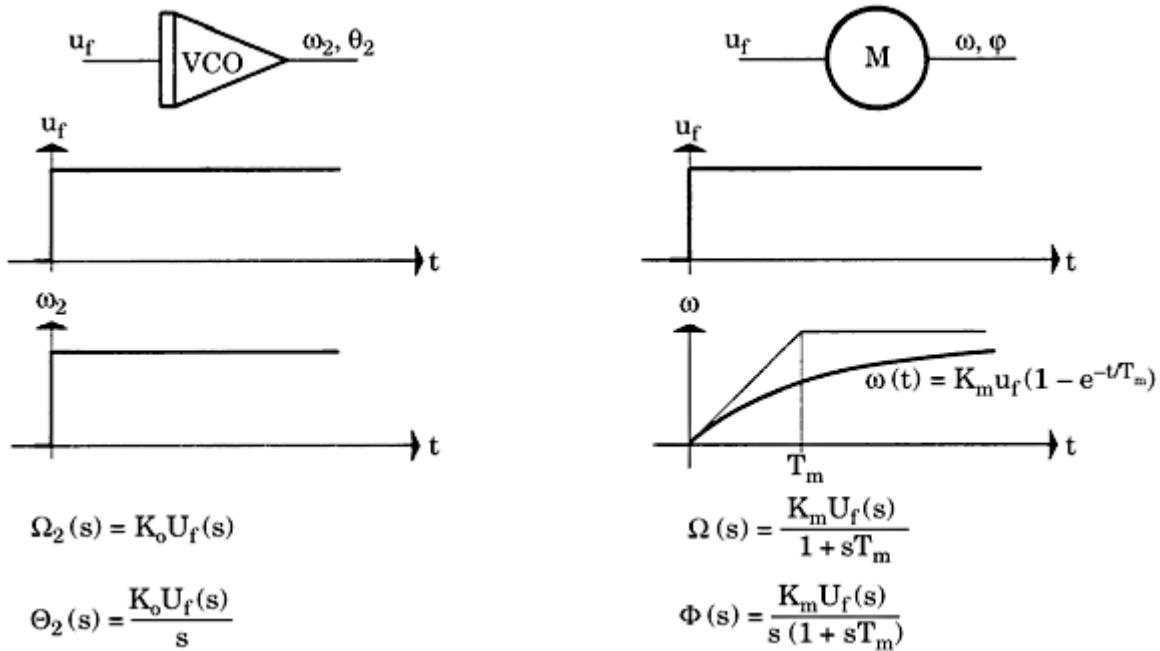
where  $K_m$  is the proportional gain and  $T_m$  is the mechanical time constant of the motor. The step response of the motor is plotted on the right in Fig. 8.10.



**Figure 8.9** The circuit of the optical tachometer generator shown in [Fig. 8.8](#).

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.



**Figure 8.10** The step response of an ordinary VCO compared with that of a motor. Note that the VCO is a first-order system, while the motor is a second-order system.

[Equation \(8.1\)](#) indicates that  $\omega$  will settle at a value proportional to  $u_f$  after some time. Applying the Laplace transform to [Eq. \(8.1\)](#) yields

$$\Omega(s) = U_f(s) \frac{K_m}{1 + sT_m} \quad (8.2)$$

The phase angle  $\phi$  of the motor is the time integral of the angular speed  $\omega$ . Therefore, we get for its Laplace transform  $\Phi(s)$ , the expression

$$\Phi(s) = U_f(s) \frac{K_m}{s(1 + sT_m)} \quad (8.3)$$

The sector disk shown in [Fig. 8.8](#) has  $K_Z$  teeth. This implies that the phase of the tachometer signal is equal to phase  $\phi$  multiplied by  $K_Z$ . Consequently, we obtain for  $\Theta_2(s)$

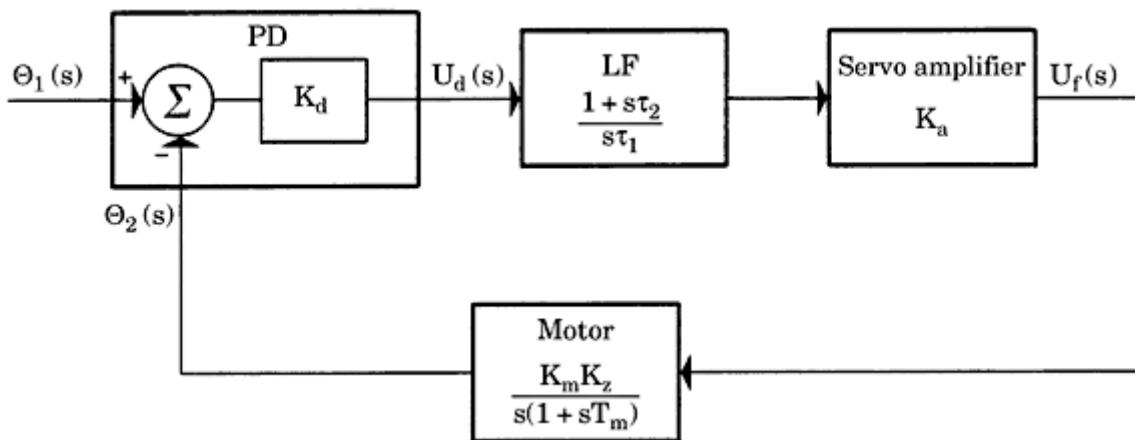
$$\Theta_2(s) = \frac{K_m K_Z}{s(1 + sT_m)} U_f(s) \quad (8.4)$$

The transfer function  $H_m(s)$  of the motor is therefore given by

$$(8.5)$$

$$H_m(s) = \frac{K_m K_Z}{s(1 + sT_m)}$$

The motor is evidently a second-order system, whereas the VCO (according to [Eq. 3.6](#)) was a first-order system only. In [Fig. 8.10](#), the transient response of

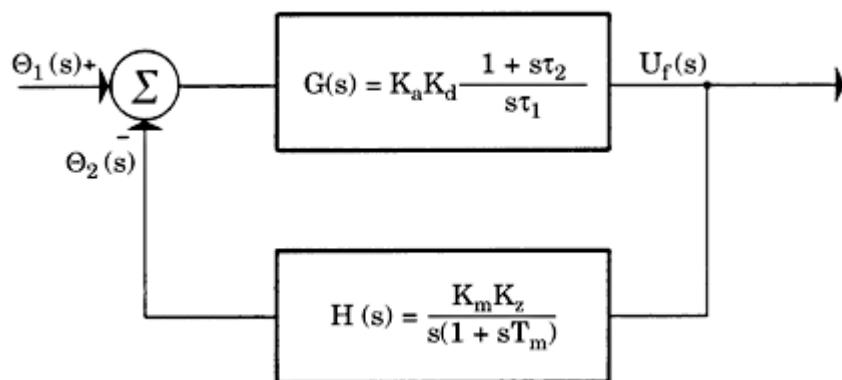


**Figure 8.11** The mathematical model of the motor-speed control system of [Fig. 8.8](#).

the motor is compared with that of an ordinary VCO. The motor-speed control system of [Fig. 8.8](#) is therefore a third-order system. The mathematical model of the control system can now be plotted ([Fig. 8.11](#)). The servo amplifier is supposed to be a zero-order gain block with proportional gain  $K_a$ . The poles of this amplifier can normally be neglected because they are at much higher frequencies than the poles of the motor. The closed system has three poles. Therefore, a filter with a zero must be specified for the loop filter; otherwise, the phase of the closed-loop transfer function would exceed  $180^\circ$  at higher frequencies and the system would become unstable. The active PI filter is chosen here for the loop filter.

The individual blocks in [Fig. 8.11](#) can be combined into fewer blocks, a result that yields the simpler block diagram of [Fig. 8.12](#). In this system, the transfer function of the forward path is defined by  $G(s)$ , whereas the transfer function of the feedback network (motor) is given by  $H(s)$ .

When a motor-speed control system is designed practically, some parameters are initially given, such as the motor parameters  $K_m$  and  $T_m$ , or the number of teeth  $K_z$  of the sector disk. The remaining parameters ( $K_a$  and  $\tau_2$ ) then have to be chosen for best dynamic performance and maximum stability of the system.



**Figure 8.12** The condensed mathematical model of the motor-speed control system.

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

This problem can be solved in many ways. It is possible to calculate the hitherto unspecified parameters by purely mathematical methods. In control engineering, however, more practical methods are preferred. To use the simplest one, we optimize PLL performance using the Bode diagram.<sup>3</sup> In the Bode diagram, amplitude and phase response of the open-loop gain of the system are plotted. From Fig. 8.13, the open-loop gain  $G_0(s)$  is given by

$$G_0(s) = G(s)H(s) = K \frac{1 + s\tau_2}{s^2(1 + sT_m)} \quad (8.6)$$

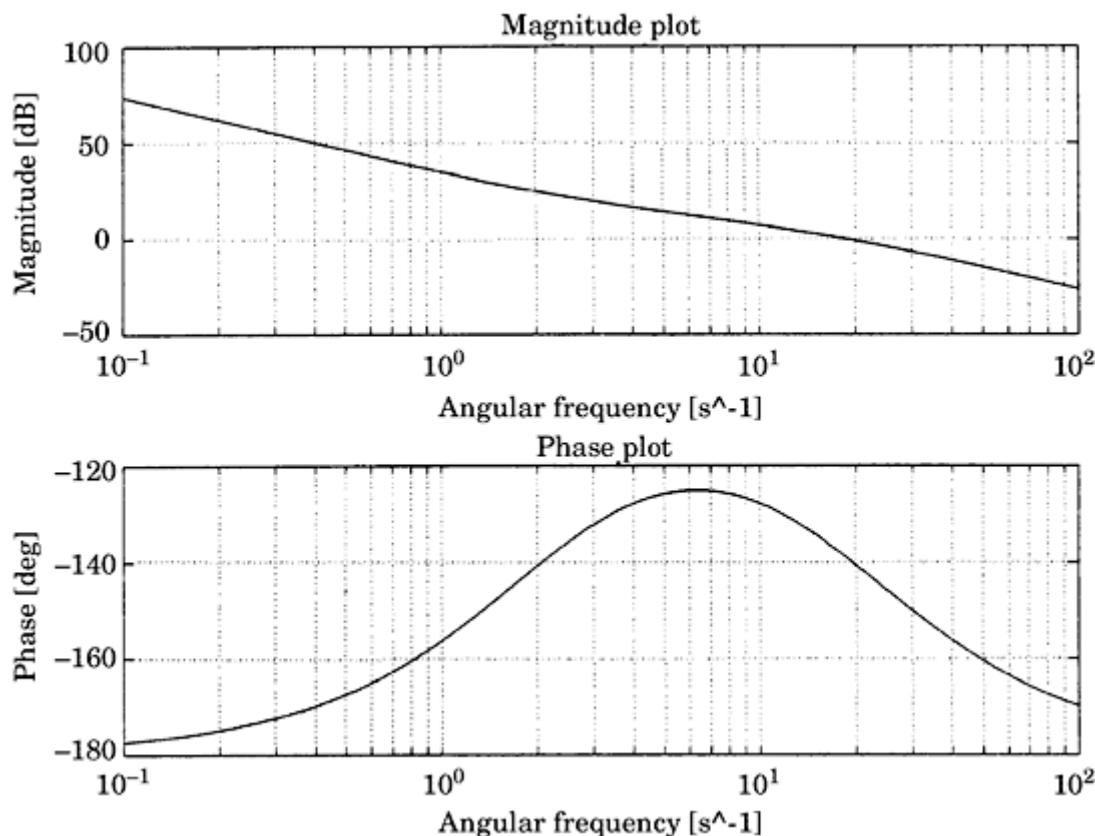
where  $K$  contains the gain factors of the individual blocks

$$K = \frac{K_d K_a K_m K_Z}{\tau_1} \quad (8.7)$$

In a practical design, phase detector gain  $K_d$ , number of teeth  $K_Z$ , motor gain  $K_m$ , and motor time constant  $T_m$  are given. The remaining parameters  $K_a$  (amplifier gain) and the two time constants  $\tau_1$  and  $\tau_2$  can be chosen freely. We assume here that parameters

$$T_m = 0.05 \text{ s}$$

$$K_m = 1$$



**Figure 8.13** Bode diagram for the open-loop gain of the motor-speed control system according to Fig. 8.8.

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

We use MathWorks Control System Toolbox<sup>26</sup> to plot and optimize the Bode diagram of our control system.<sup>25</sup> Looking at Eq. (8.6), we see that the magnitude response will start with a slope of  $-40$  dB/decade at low frequencies, due to the term  $s^2$  in the denominator. Above the angular frequency  $\omega_m = 1/T_m$ , the slope will become  $-60$  dB/decade, so at higher frequencies the phase would approach  $270^\circ$  if there were no zero in the transfer function. The zero of the transfer function must be placed such that the phase stays well below  $180^\circ$  at the frequency where the magnitude curve goes through the 0-dB line. To get acceptable stability of the loop, we require a *phase margin*<sup>3</sup> of at least  $30^\circ$ . We conclude that the break frequency of the zero term  $(1 + s\tau_2)$  must be well below the break frequency  $1/T_m$ , say by a factor of 10. Therefore,  $\tau_2$  is tentatively set to 0.5 s. Using the Control System Toolbox, we vary the overall gain such that the magnitude is just 0 dB at the break frequency  $\omega = 1/T_m$ . A value of  $K = 50$  was required to attain this goal. The resulting Bode diagram is shown in Fig. 8.13. The magnitude curve starts with  $-40$  dB/decade at low frequencies. The transfer function zero at  $\omega = 1/\tau_2 = 2$   $s^{-1}$  causes the slope to change to  $-20$  dB/decade—meaning the magnitude curve “bends up.” This causes the phase curve to increase from  $-180^\circ$  toward  $-90^\circ$ , as shown in the phase plot. At the break frequency  $\omega = 1/T_m = 20$   $s^{-1}$ , the slope of the magnitude curve becomes  $-40$  dB/decade again. The magnitude crosses the 0-dB line at about  $\omega = 20$   $s^{-1}$ , and the phase margin is about  $40^\circ$ , which is sufficient for good transient response. Knowing total gain  $K$ , the amplifier gain  $K_a$  and the remaining time constant  $\tau_1$  can be specified as soon as the number of teeth  $K_Z$  is given.

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

# Higher-Order Loops

## Motivation for Higher-Order Loops

In Chaps. 2 and 3, we considered mainly second-order PLLs. These loops used first-order loop filters as shown in Figs. 2.17, 2.19, and 2.21. As can be seen from the Bode diagrams of the loop filter transfer functions in Figs. 2.18, 2.20, and 2.22 the gain rolls off at  $-20$  dB/decade at higher frequencies but asymptotically approaches a nonzero value for radian frequencies higher than  $1/\tau_2$ , where  $\tau_2$  is the time constant in the numerator of the filter transfer function (cf. Eqs. 2.29 through 2.31).

In the discussion of spectral purity of PLL frequency synthesizers, we recognized that reference frequency feedthrough becomes a problem. Spurious side-bands can be intolerable if the loop filter does not sufficiently attenuate AC components at the reference frequency (plus harmonics) that are created by the phase detector. To reduce reference frequency feedthrough, we must use higher-order loop filters—in other words, loop filters of order 2 or higher.

With higher-order loop filters, loop stability becomes an issue. Getting stable operation with a second-order PLL was easy because the open-loop transfer function had two poles and one zero. A pole creates a phase shift of  $-90^\circ$  at higher frequencies, and a zero creates a phase shift of  $+90^\circ$ . When the poles and the zero are properly located, the overall phase shift never comes close to  $-180^\circ$ ; hence, the loop stays stable. This goal was easily met by choosing time constant  $\tau_2$  of the loop filter such that a reasonable damping factor  $\zeta$  was obtained. If the loop filter has two or more poles, the phase shift can become larger than  $180^\circ$ , hence the poles and zeroes of the loop filter must be placed such that stability is maintained. We will deal with the stability of loops in the following section.

## Analyzing the Stability of Higher-Order Loops

Control theory offers many mathematical methods for analyzing stability of feedback systems. Some are sophisticated and mathematically demanding, but there is also a simple one: the Bode plot. Drawing a Bode plot is very easy for

systems like the PLL; we will demonstrate that by the example of a third-order PLL. The corresponding loop filter has order two. Assuming a passive lead-lag, the filter transfer function becomes

$$F(s) = \frac{1 + sT_2}{(1 + sT_1)(1 + sT_3)} \quad (9.1)$$

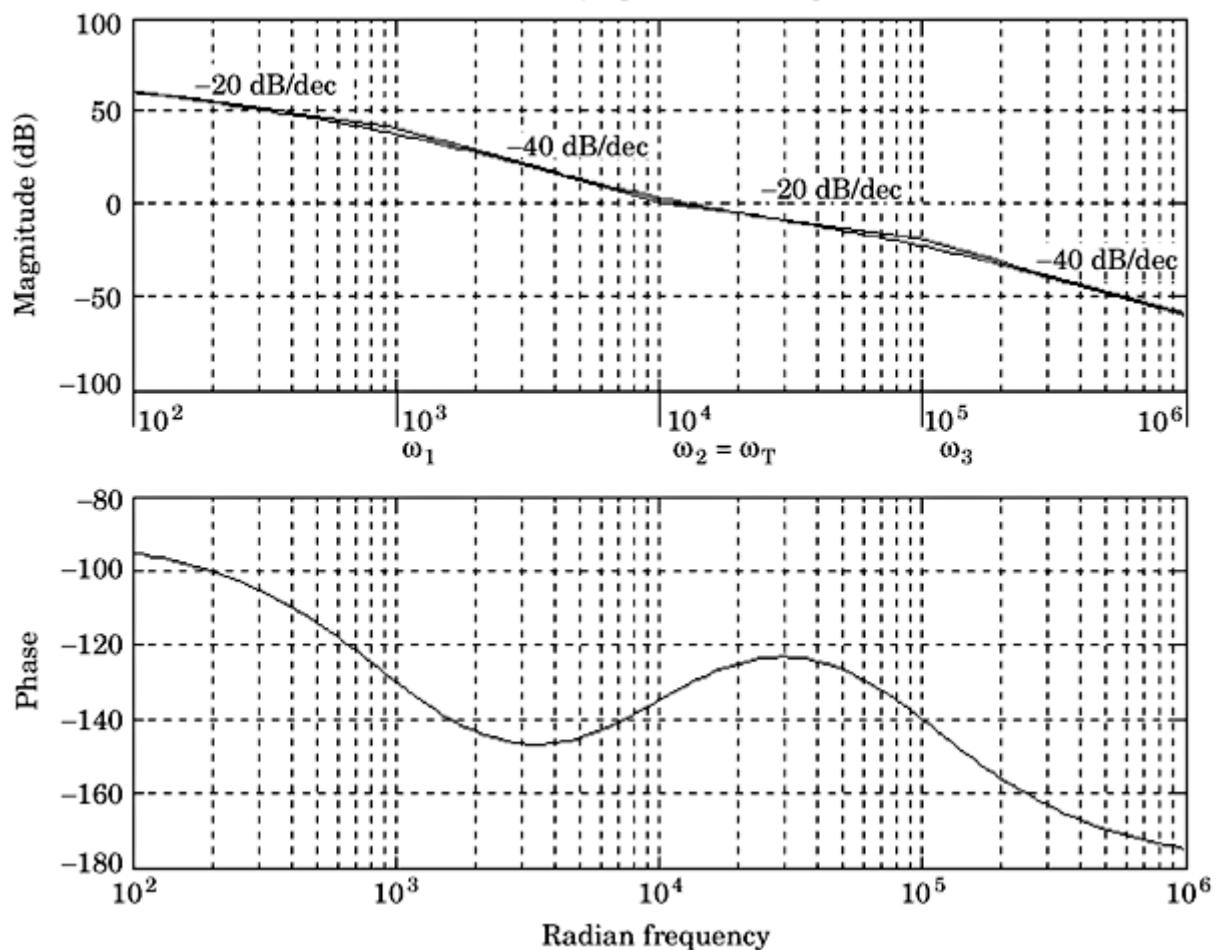
where  $T_1$ ,  $T_2$ , and  $T_3$  are time constants, and it is assumed that  $T_1 > T_2 > T_3$ . The open-loop transfer function of a PLL built with this filter is given by

$$G(s) = \frac{K_0 K_d}{N} \cdot \frac{(1 + sT_2)}{s(1 + sT_1)(1 + sT_3)} \quad (9.2)$$

(also refer to the PLL model in [Fig. 3.1](#)).

To analyze stability, the Bode diagram for  $G(s)$  is plotted. This is done by replacing  $s$  with  $j\omega$  and plotting the magnitude and phase of  $G(\omega)$  versus radian frequency  $\omega$ . The magnitude plot is logarithmic on both axes, while the phase plot is logarithmic on the frequency axis, but linear on the phase axis. The Bode plot is shown in [Fig. 9.1](#). In the magnitude plot, the exact and the asymptotic

### Exact and asymptotic bode diagrams



**Figure 9.1** Analyzing loop stability using the Bode diagram.

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
 Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
 Any use is subject to the Terms of Use as given at the website.

diagram are shown. The asymptotic curve consists of line segments and is an approximation to the exact magnitude function. The magnitude plot has three corners at radian frequencies  $\omega_1 = 1/T_1$ ,  $\omega_2 = 1/T_2$ , and  $\omega_3 = 1/T_3$ . The corners at  $\omega_1$  and  $\omega_3$  are created by the poles of the transfer function, and the corner at  $\omega_2$  by its zero. At low frequencies the magnitude is dominated by the term  $1/s$ . Because the magnitude (gain) rolls off with  $1/\omega$ , the slope of the magnitude curve is  $-20$  dB/decade (dec = decade of frequency). Above  $\omega = \omega_1$ , the first pole of the loop filter comes into play; now the asymptotic value of the slope is increased to  $-40$  dB/decade. At  $\omega = \omega_2$ , the zero of the filter becomes active. The term in the numerator of Eq. (4.2) causes the slope of the magnitude curve to bend up toward  $-20$  dB/decade again. Above  $\omega = \omega_3$ , however, the slope becomes  $-40$  dB/decade due to the second pole of the loop filter.

Now let's look at the phase plot. For minimum phase networks, the phase plot and magnitude plot are closely related. (A system is minimum phase if all the poles and zeroes of its transfer function are in the left half of the  $s$ -plane; this is the case for all practical PLLs.) Under this condition, the phase approaches an asymptotic value of  $-90^\circ$  when the slope of the magnitude plot is  $-20$  dB/decade. The phase plot asymptotically approaches  $-180^\circ$  if the magnitude slope is  $-40$  dB/decade, and so on. We see from the phase plot that the curve starts at  $-90^\circ$  at low frequencies. Above  $\omega = \omega_1$ , the phase bends down toward  $-180^\circ$ . Above  $\omega = \omega_2$ , however, the asymptotic value of phase becomes  $-90^\circ$  due to the onset of the filters zero. Finally, above  $\omega = \omega_3$ , the asymptotic value of the phase becomes  $-180^\circ$  again. We note that the phase curve exhibits a "peak" between frequencies  $\omega_2$  and  $\omega_3$  and therefore has a local maximum at some distinct frequency within this range.

The magnitude curve crosses the 0-dB line at a radian frequency named *transition frequency*  $\omega_T$ . At the transition frequency, the open-loop gain is exactly 1. The system is stable if the *phase of  $G(\omega_T)$  is more positive than  $-180^\circ$* . Let's denote that phase value by  $\varphi(\omega_T)$ . The quantity  $180^\circ + \varphi(\omega_T)$  is called *phase margin*  $\varphi_m$ . In control engineering, one is attempted to get phase margins between about  $30 - 60^\circ$ . When the phase margin is very low (a few degrees only) the system is stable but heavily underdamped, hence its transient response is oscillatory and dies out slowly. When the phase margin is too large, the dynamic response becomes aperiodic and can get sluggish. In the example of Fig. 9.1, the corner frequency  $\omega_2$  was purposely chosen so as to coincide with the transition frequency  $\omega_T$ . Under this condition, the phase margin is approximately  $45^\circ$ , which is sufficient to obtain fair stability.

The example in Fig. 9.1 was related to a third-order PLL. When building PLL with an order higher than 3, additional poles (and eventually zeroes) must be placed. In the case of a fifth-order PLL, for example, the open-loop transfer function  $G(s)$  has five poles and may also have up to five zeroes. The gain rolloff at higher frequencies has a slope that is given by

$$S = 20(n_z - n_p) \text{ dB} \quad (9.3)$$

where  $n_z$  = number of zeroes, and  $n_p$  = number of poles of  $G(s)$ . Because we want the gain to rolloff with maximum slope, the number of zeroes should be chosen as small as possible. In most higher-order PLL designs, only one zero is chosen. This zero is placed exactly as demonstrated in the example in Fig. 9.1. If the system has a higher order than 3, additional poles must be placed at frequencies above  $\omega_3$  in Fig. 9.1. This will be described in the following sections.

As we have seen, the poles and the zero of the system have been placed with reference to the transition frequency  $\omega_T$ . Consequently, the loop design should start with an initial value for  $\omega_T$ . Usually, the designer of a PLL has an idea about the desired loop bandwidth. A number of parameters are related to bandwidth—for example,  $\omega_{3dB}$  [the 3-dB closed-loop bandwidth of  $H(\omega)$ ],  $B_L$  (the noise bandwidth of the loop), or  $\omega_n$  (the natural frequency). All these parameters, including  $\omega_T$ , are close together—in other words, they have about the same size. Because the natural frequency is only defined for second-order systems (in a strict sense), it seems more practical to start by setting  $\omega_{3dB}$ . Often,  $\omega_{3dB}$  is chosen to be around 1/20 the (scaled-down) center frequency  $\omega_0'$ . What we need now is a mathematical relation between  $\omega_{3dB}$  and  $\omega_T$ . For a second-order high-gain loop with  $\zeta = 0.707$ , it can be shown that  $\omega_{3dB} \approx 2.06 \omega_n$  and  $\omega_T \approx 1.55 \omega_n$ . Hence, we have

$$\omega_T \approx \omega_{3dB}/1.33 \quad (9.4)$$

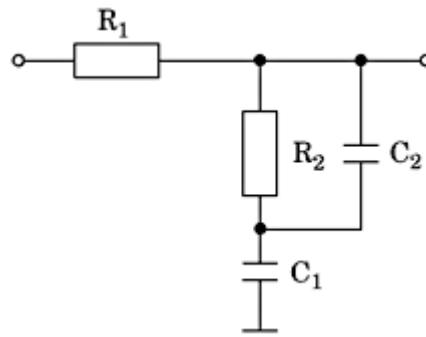
Experience has shown that this ratio stays nearly unchanged for higher-order PLLs. This is a consequence of the fact that we always try to shape the magnitude plot such that its magnitude curve crosses the 0-dB line with a slope of about  $-20$  dB/decade. The poles at higher frequencies than  $\omega_T$  then have little influence on the Bode plot in the vicinity of  $\omega_T$ , hence the ratio  $\omega_{3dB}/\omega_T$  is mostly in the range of 1.3 to 1.5.

In the following sections, we will develop procedures for the design of loop filters for PLLs having order 3 to 5. All of these methods can be automated (in Chap. 10, a computer program will be presented that allows you to design mixed-signal PLLs up to order 5 [and all-digital PLLs] automatically). This is a software tool developed by the author and is distributed with the book.

## Designing Third-Order PLLs

In a third-order PLL, a second-order loop filter must be chosen. In this section, we describe four variants of a second-order loop filter, a passive lead-lag filter for use with a voltage output phase detector, a passive lead-lag filter for use with a current-output phase detector, an active lead-lag, and an active PI filter. The last two filters are applied uniquely in conjunction with voltage output phase detectors, hence we will consider only active filter types for voltage input. We start the discussion with the passive lead-lag.

**Any use is subject to the Terms of Use as given at the website.**



**Figure 9.2** A second-order passive lead-lag loop filter.

### The passive lead-lag loop filter for voltage input

The schematic of this filter is shown in [Fig. 9.2](#). Its transfer function is

$$F(s) = \frac{1 + s(\tau_2 + \tau_3)}{1 + s(\tau_1 + \tau_2 + \tau_3) + s^2\tau_1\tau_3} \quad (9.5)$$

with  $\tau_1 = R_1 C_1$ ,  $\tau_2 = R_2 C_1$ , and  $\tau_3 = R_2 C_2$ . To place the poles and zeroes of the PLL's open-loop transfer function  $G(s)$ , it is more convenient to write this function in factorized form:

$$F(s) = \frac{1 + sT_2}{(1 + sT_1)(1 + sT_3)} \quad (9.6)$$

The open-loop transfer function of the third-order PLL then becomes

$$G(s) = \frac{K_0 K_d}{Ns} \cdot \frac{1 + sT_2}{(1 + sT_1)(1 + sT_3)} \quad (9.7)$$

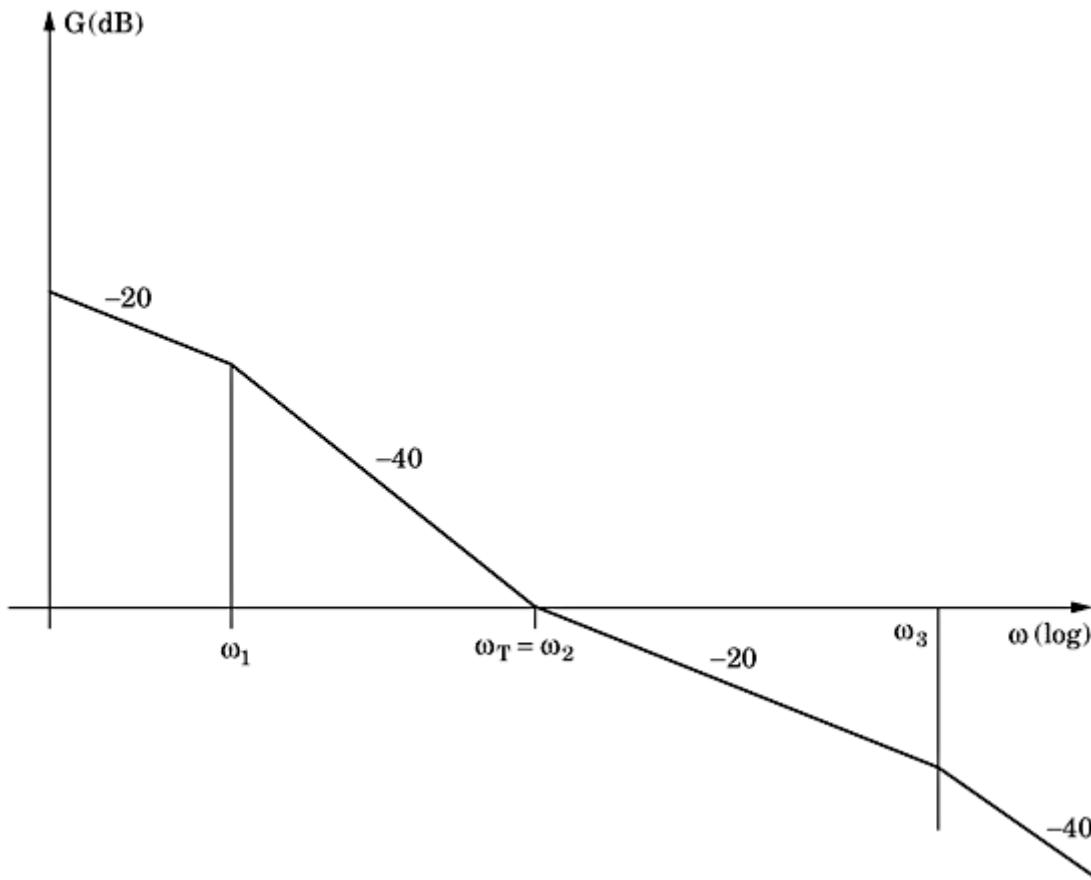
To get the Bode diagram, we set  $s = j\omega$  and plot the magnitude  $|G(\omega)|$  versus the radian frequency  $\omega$ . This is shown in [Fig. 9.3](#). To determine the corner frequencies  $\omega_1$ ,  $\omega_2$ , and  $\omega_3$ , the transition frequency  $\omega_T$  must be known first. It has proven most convenient to proceed as described in the following steps:

**Step 1** Normally the designer knows what should be the bandwidth of the PLL, hence it is reasonable to start by choosing a value for  $\omega_{3dB}$ . A default value could be  $\omega_{3dB} = 0.05 \cdot \omega_0'$ , where  $\omega_0'$  is the down-scaled center (radian) frequency of the PLL.

**Step 2** Using the approximation in [Eq. \(9.4\)](#), we compute  $\omega_T = \omega_{3dB}/1.33$ . With the passive lead-lag filter, there is an upper limit for  $\omega_T$ , however, which is given by

$$\omega_{T,\max} = \frac{K_0 K_d}{N} \quad (9.8)$$

When a larger value of  $\omega_T$  is specified, this later results in a negative value for  $\tau_1$  (cf. [Eq. 9.5](#)), hence that filter would be unrealizable.



**Figure 9.3** The Bode diagram (magnitude plot) of a third-order PLL using the passive lead-lag loop filter.

**Step 3** The corner frequencies  $\omega_2$  and  $\omega_3$  are now chosen. Following the procedure presented in [Sec. 9.2](#), it is recommended to set  $\omega_2 = \omega_T$ , which results in a phase margin of around  $45^\circ$ . Then, we set  $\omega_3 = 5\omega_2$ . Theoretically,  $\omega_3$  could be chosen closer to  $\omega_2$ , but when it comes too close, the slope of the magnitude curve between  $\omega_2$  and  $\omega_3$  becomes steeper, which results in a poorer phase margin. Setting  $\omega_3 = 5\omega_2$  is a fair compromise.

**Step 4** Now the corner frequency  $\omega_1$  must be chosen such that the open-loop gain at the transition frequency  $\omega_T$  becomes 1. This leads to

$$\omega_1 = \frac{\omega_T^2 N}{K_0 K_d} \quad (9.9)$$

**Step 5** The time constants  $T_1$ ,  $T_2$ , and  $T_3$  are computed from

$$T_1 = 1/\omega_1, T_2 = 1/\omega_2, \text{ and } T_3 = 1/\omega_3.$$

**Step 6** Given  $T_1$  through  $T_3$ , we can calculate the filter time constants  $\tau_1$  through  $\tau_3$  now. By comparison of coefficients in Eqs. (9.5) and (9.6), we get

$$\begin{aligned}\tau_1 &= T_1 + T_3 - T_2 \\ \tau_2 &= T_2 - \frac{T_1 T_3}{T_1 + T_3 - T_2} \\ \tau_3 &= T_2 - \tau_2\end{aligned}\tag{9.10}$$

**Step 7** (Optional) There are cases where the designer modifies the values for  $\tau_1$  through  $\tau_3$  or adopts the values for  $\tau_1$  through  $\tau_3$  from an earlier design. When these parameters have been directly set or altered, we certainly want to know what is the effect on the quantities  $T_1$  through  $T_3$ —in other words, onto the corner frequencies  $\omega_1$  through  $\omega_3$  in the magnitude plot. When solving the Eqs. (9.10) for  $T_1$  through  $T_3$ , we get

$$T_1 = \frac{\tau_1 + \tau_2 + \tau_3 + \sqrt{(\tau_1 + \tau_2 + \tau_3)^2 - 4\tau_1\tau_3}}{2}\tag{9.11}$$

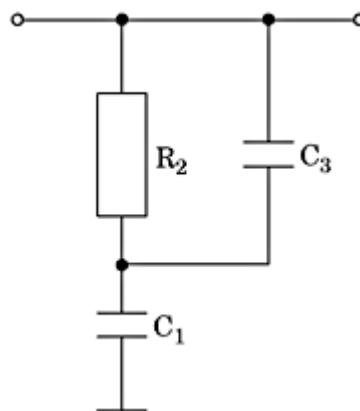
$$T_2 = \tau_2 + \tau_3$$

$$T_3 = \frac{\tau_1\tau_3}{T_1}$$

**Step 8** Finally, the filter components ( $R$ ,  $C$ ) are computed from the values for  $\tau_1$  through  $\tau_3$  [cf. Eq. (9.5)]. The value of  $C_1$  can be chosen arbitrarily. It should be set such that “reasonable” values for the resistors are obtained—in other words, in the region of about 1 through 100 k $\Omega$ .

### Passive lead-lag loop filter for current input

The schematic of the passive lead-lag filter for current input (charge pump) is shown in Fig. 9.4.



**Figure 9.4** The passive lead-lag filter for use with phase detectors having current output.

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

The open-loop transfer function of the third-order PLL then becomes

$$G(s) = \frac{K_P K_0}{s^2 C_1 N} \cdot \frac{1 + s(\tau_2 + \tau_3)}{1 + s\tau_3} \quad (9.12)$$

with  $\tau_2 = R_2 C_1$  and  $\tau_3 = R_2 C_3$ .

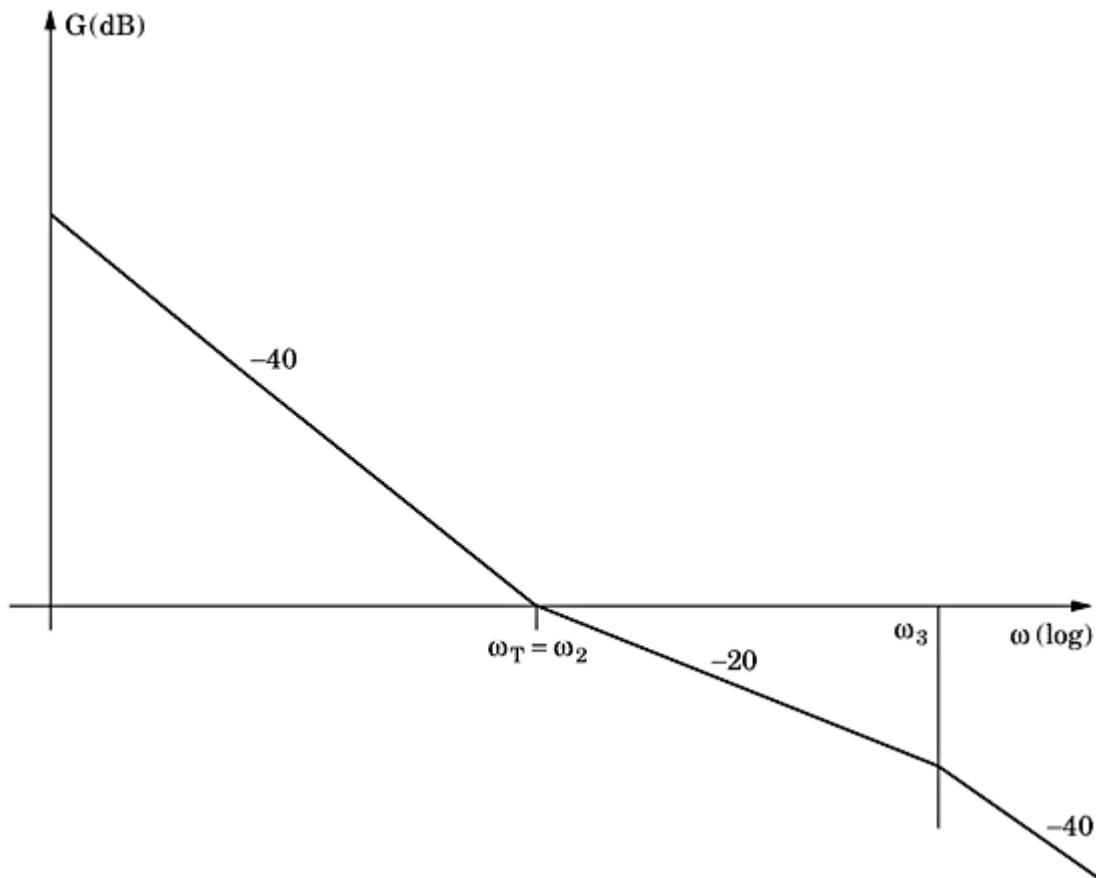
To place the poles and zeroes of the PLL's open-loop transfer function  $G(s)$ , it is more convenient to write this function in factorized form:

$$G(s) = \frac{K_P K_0}{s^2 C_1 N} \cdot \frac{1 + sT_2}{1 + sT_3} \quad (9.13)$$

The magnitude plot of this system is shown in Fig. 9.5.

To determine the corner frequencies  $\omega_2$  and  $\omega_3$ , the transition frequency  $\omega_T$  must be known first. It has proven most convenient to proceed as described in the following steps:

**Step 1** Normally, the designer knows what should be the bandwidth of the PLL, hence it is reasonable to start by choosing a value for  $\omega_{3dB}$ . A default value could be  $\omega_{3dB} = 0.05 \cdot \omega_0'$ , where  $\omega_0'$  is the down-scaled center (radian) frequency of the PLL.



**Figure 9.5** The magnitude plot of open-loop gain of a third-order PLL using a charge pump

PFD and a passive lead-lag loop filter.

---

**Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.**

**Step 2** Using the approximation in Eq. (9.4), we compute  $\omega_T = \omega_{3\text{dB}}/1.33$ .

**Step 3** The corner frequency  $\omega_2$  is chosen to coincide with  $\omega_T$ , which results in a phase margin of around  $45^\circ$ . Then, we set  $\omega_3 = 5\omega_2$ , as done in the previous example (Sec. 9.3.1).

**Step 4** The capacitor  $C_1$  must be chosen such that the magnitude of the open-loop gain becomes 1 at the transition frequency. This results in

$$C_1 = \frac{K_0 K_P}{N \omega_T^2} \quad (9.14)$$

**Step 5** The time constants  $T_2$  and  $T_3$  are computed from

$$T_2 = 1/\omega_2 \text{ and } T_3 = 1/\omega_3.$$

**Step 6** Given  $T_2$  and  $T_3$ , we can now calculate the filter time constants  $\tau_2$  and  $\tau_3$ . By the comparison of coefficients in Eqs. (9.12) and (9.13), we get

$$\begin{aligned} \tau_2 &= T_2 - T_3 \\ \tau_3 &= T_3 \end{aligned} \quad (9.15)$$

**Step 7** (Optional) There are cases where the designer modifies the values for  $\tau_2$  and  $\tau_3$  or adopts the values for  $\tau_2$  and  $\tau_3$  from an earlier design. When these parameters have been directly set or altered, we certainly want to know what the effect is on the quantities  $T_2$  and  $T_3$ —in other words, on the corner frequencies  $\omega_2$  and  $\omega_3$  in the magnitude plot. When solving the Eqs. (9.15) for  $T_2$  and  $T_3$ , we get

$$\begin{aligned} T_3 &= \tau_3 \\ T_2 &= \tau_2 + \tau_3 \end{aligned} \quad (9.16)$$

**Step 8** Finally, the filter components ( $R, C$ ) are computed from the values for  $\tau_2$  and  $\tau_3$  [cf. Eq. (9.12)].

### Active lead-lag loop filter

The schematic of a second-order active lead-lag filter is shown in Fig. 9.6.

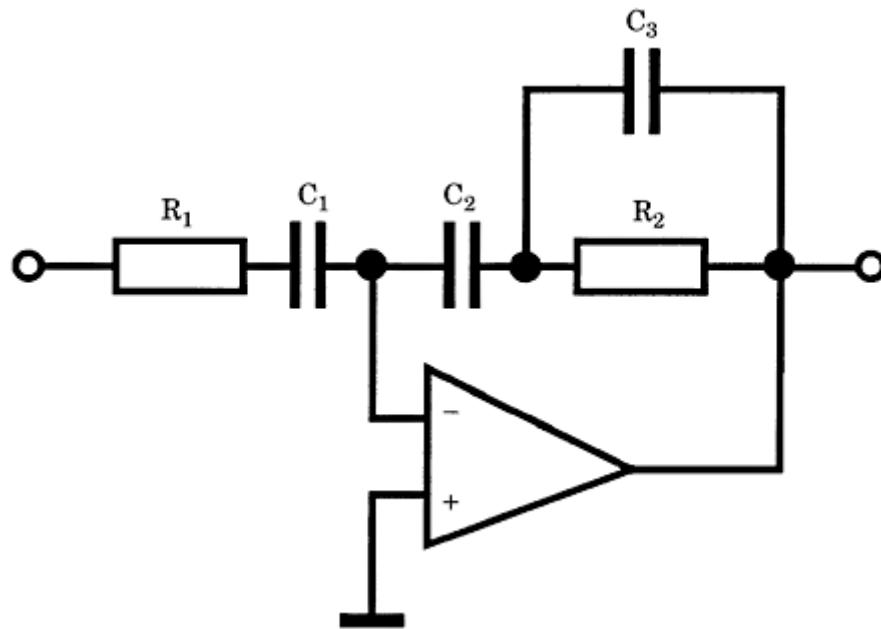
The transfer function for this filter is given by

$$F(s) = K_a \frac{1 + s(\tau_2 + \tau_3)}{(1 + s\tau_1)(1 + s\tau_3)} \quad (9.17)$$

with  $\tau_1 = R_1 C_1$ ,  $\tau_2 = R_2 C_2$ ,  $\tau_3 = R_2 C_3$ , and  $K_a = C_1/C_2$ . Because we must determine the corner

frequencies of the Bode plot, it is more convenient to write  $F(s)$  in the standardized form as

$$F(s) = K_a \frac{1 + sT_2}{(1 + sT_1)(1 + sT_3)} \quad (9.18)$$



**Figure 9.6** A schematic of the second-order active lead-lag filter.

The open-loop gain of the third-order PLL then reads

$$G(s) = \frac{K_0 K_d K_a}{Ns} \cdot \frac{1 + sT_2}{(1 + sT_1)(1 + sT_3)} \quad (9.19)$$

For this design, the procedure is very similar to that for the passive lead-lag filter. The Bode plot in [Fig. 9.3](#) remains valid for this type of loop filter. The design steps are as follows:

**Step 1** Normally, the designer knows what should be the bandwidth of the PLL, hence it is reasonable to start by choosing a value for  $\omega_{3\text{dB}}$ . A default value could be  $\omega_{3\text{dB}} = 0.05 \cdot \omega_0'$ , where  $\omega_0'$  is the down-scaled center (radian) frequency of the PLL.

**Step 2** Using the approximation in [Eq. \(9.4\)](#), we compute  $\omega_T = \omega_{3\text{dB}}/1.33$ . In contrast to the passive lead-lag filter with voltage input, there is no mathematical restriction on the size of  $\omega_T$  here, because we have an additional parameter  $K_a$ , which can always be specified such that the open-loop gain becomes 1 at the transition frequency  $\omega_T$ .

**Step 3** The corner frequencies  $\omega_2$  and  $\omega_3$  are now chosen. We recommend choosing  $\omega_2 = \omega_T$ , as explained in [Sec. 9.2](#). Setting  $\omega_3 = 5\omega_2$  again yields fair loop stability.

**Step 4** It is recommended to set  $\omega_1 = \omega_2/10$ , thus the open-loop gain rolls off with  $-40$  dB/decade over one full decade of frequency.  $K_a$  must be chosen such that the open-loop gain becomes 1 at  $\omega_T$ . For  $K_a$ , we get

$$K_a = \frac{10N\omega_T}{K_0 K_d} \quad (9.20)$$

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

**Step 5** The time constants  $T_1$ ,  $T_2$ , and  $T_3$  are computed from

$$T_1 = 1/\omega_1, T_2 = 1/\omega_2, \text{ and } T_3 = 1/\omega_3$$

**Step 6** Given  $T_1$  through  $T_3$ , we can now calculate the filter time constants  $\tau_1$  through  $\tau_3$ . By comparison of the coefficients in Eqs. (9.17) and (9.18), we get

$$\begin{aligned}\tau_1 &= T_1 \\ \tau_2 &= T_2 - T_3 \\ \tau_3 &= T_3\end{aligned}\tag{9.21}$$

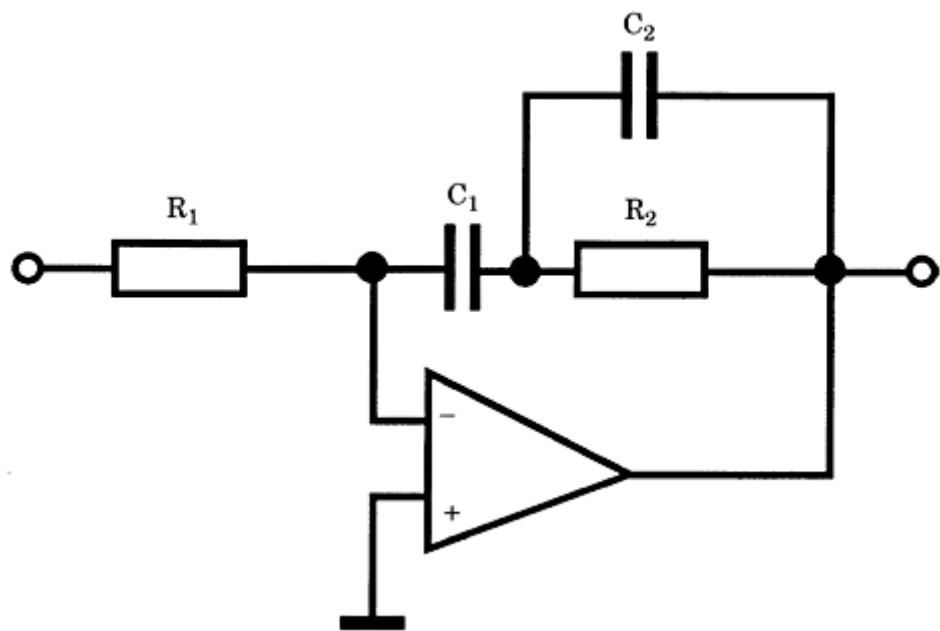
**Step 7** (Optional) There are cases where the designer modifies the values for  $\tau_1$  through  $\tau_3$  or adopts the values for  $\tau_1$  through  $\tau_3$  from an earlier design. When these parameters have been directly set or altered, it may be desirable to also know the values for  $T_1$  through  $T_3$ —meaning, the corner frequencies  $\omega_1$  through  $\omega_3$  in the magnitude plot. When solving the Eqs. (9.21) for  $T_1$  through  $T_3$ , we get

$$\begin{aligned}T_1 &= \tau_1 \\ T_2 &= \tau_2 + \tau_3 \\ T_3 &= \tau_3\end{aligned}\tag{9.22}$$

**Step 8** Finally, the filter components ( $R$ ,  $C$ ) are computed from the values for  $\tau_1$  through  $\tau_3$ . The value of  $C_1$  can be chosen arbitrarily. It should be set such that “reasonable” values for the resistors are obtained—in other words, in the region of about 1 through 100 k $\Omega$ .

### Active PI loop filter

The schematic of a second-order active PI filter is shown in [Fig. 9.7](#).



**Figure 9.7** A schematic of a second-order active PI loop filter.

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

The transfer function of this filter is given by

$$F(s) = \frac{1 + s(\tau_2 + \tau_3)}{s\tau_1(1 + s\tau_3)} \quad (9.23)$$

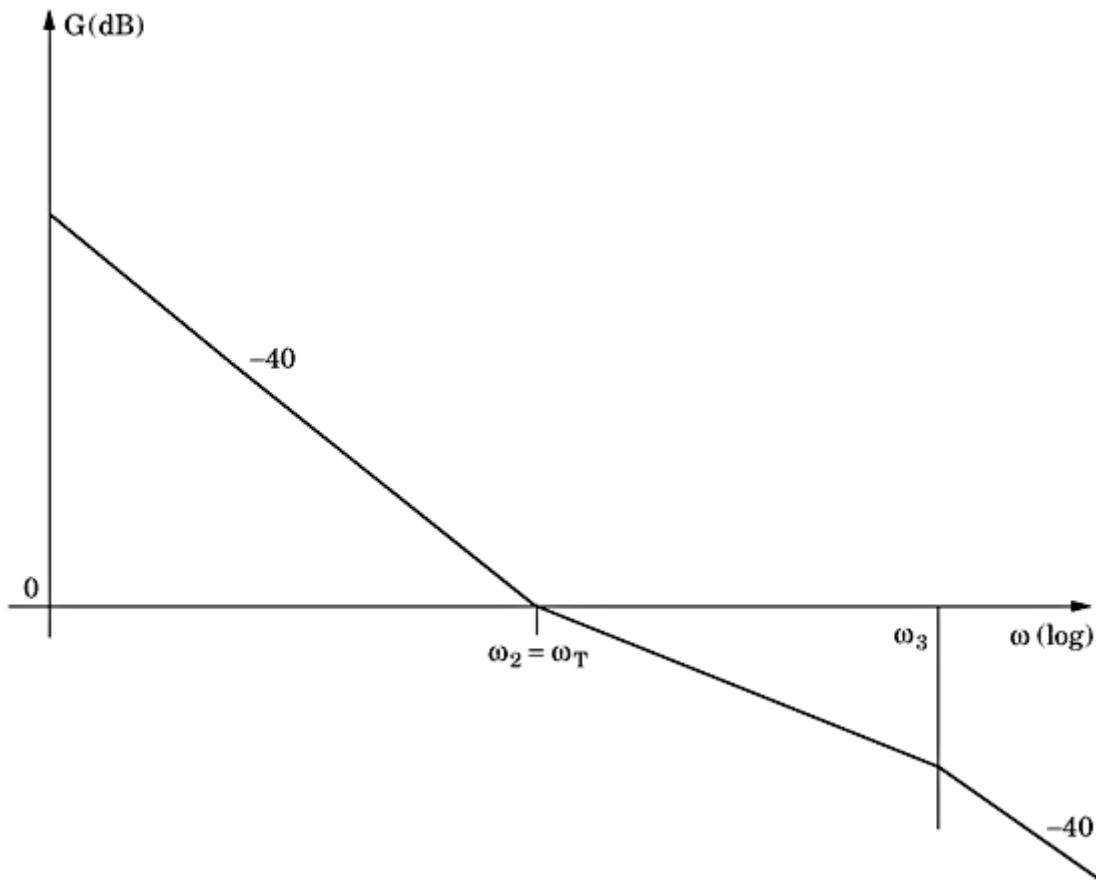
with  $\tau_1 = R_1C_1$ ,  $\tau_2 = R_2C_1$ , and  $\tau_3 = R_2C_2$ . Because we must determine the corner frequencies of the Bode plot, it is more convenient to write  $F(s)$  in the standardized form as

$$F(s) = \frac{1 + sT_2}{sT_1(1 + sT_3)} \quad (9.24)$$

The open-loop gain of the third-order PLL then reads

$$G(s) = \frac{K_0 K_d}{Ns} \cdot \frac{1 + sT_2}{sT_1(1 + sT_3)} \quad (9.25)$$

The Bode plot (magnitude) for this system is shown in [Fig. 9.8](#). For this design, the procedure is very similar to that of the active lead-lag filter. The design steps are as follows:



**Figure 9.8** The Bode plot of a third-order PLL using the active PI loop filter.

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

**Step 1** Normally, the designer knows what should be the bandwidth of the PLL, hence it is reasonable to start by choosing a value for  $\omega_{3\text{dB}}$ . A default value could be  $\omega_{3\text{dB}} = 0.05 \cdot \omega_0'$ , where  $\omega_0'$  is the down-scaled center (radian) frequency of the PLL.

**Step 2** Using the approximation in [Eq. \(9.4\)](#), we compute  $\omega_T = \omega_{3\text{dB}}/1.33$ . There is no mathematical restriction on the size of  $\omega_T$  because the parameter  $T_1$  can be chosen such that the open-loop gain becomes 1 at  $\omega = \omega_T$ .

**Step 3** The corner frequencies  $\omega_2$  and  $\omega_3$  are now chosen. To get sufficient phase margin, it is recommended to set  $\omega_2 = \omega_T$ , as explained in [Sec. 9.2](#). Then, for good loop stability, we recommend setting  $\omega_3 = 5\omega_2$ .

**Step 4** Parameter  $T_1$  is chosen such that the open-loop gain is 1 at  $\omega = \omega_T$ . This leads to

$$T_1 = \frac{K_0 K_d}{N \omega_T^2} \quad (9.26)$$

**Step 5** The time constants  $T_2$  and  $T_3$  are computed from

$$T_2 = 1/\omega_2 \quad \text{and} \quad T_3 = 1/\omega_3$$

**Step 6** Given  $T_1$  through  $T_3$ , we can now calculate the filter time constants  $\tau_1$  through  $\tau_3$ . By comparing the coefficients in Eqs. [\(9.23\)](#) and [\(9.24\)](#), we get

$$\begin{aligned} \tau_1 &= T_1 \\ \tau_2 &= T_2 - T_3 \\ \tau_3 &= T_3 \end{aligned} \quad (9.27)$$

**Step 7 (Optional)** There are cases where the designer modifies the values for  $\tau_1$  through  $\tau_3$  or adopts the values for  $\tau_1$  through  $\tau_3$  from an earlier design. When these parameters have been directly set or altered, it becomes desirable to know the effect on the values for  $T_1$  through  $T_3$ —in other words, on the corner frequencies  $\omega_1$  through  $\omega_3$  in the magnitude plot. When solving the Eqs. [\(9.27\)](#) for  $T_1$  through  $T_3$ , we get

$$\begin{aligned} T_1 &= \tau_1 \\ T_2 &= \tau_2 + \tau_3 \\ T_3 &= \tau_3 \end{aligned} \quad (9.28)$$

**Step 8** Finally, the filter components ( $R, C$ ) are computed from the values for  $\tau_1$  through  $\tau_3$ . The value of  $C_1$  can be chosen arbitrarily. It should be set such that “reasonable” values for

the resistors are obtained—in other words, in the region of about 1 through 100 k $\Omega$ .

## Designing Fourth-Order PLLs

To obtain a fourth-order PLL, we must use a third-order loop filter. Such a filter can be realized again as a passive lead-lag, an active lead-lag, or an active PI filter. All these filters will have three poles. When a passive RC filter is implemented, all the poles are real (thus, in the complex  $s$ -plane they are on the negative real axis). To get complex-conjugate pole pairs, we would also have to use inductors, but most designers try to avoid them if possible. When building an active RC filter, it is possible to realize complex-conjugate pole pairs. In the following design, procedures are given for four types of third-order loop filters: the passive lead-lag loop filter with voltage input, the passive lead-lag loop filter with current input, the active lead-lag, and the active PI filter. The latter two will be used exclusively with a voltage drive.

### Passive lead-lag loop filters for voltage input

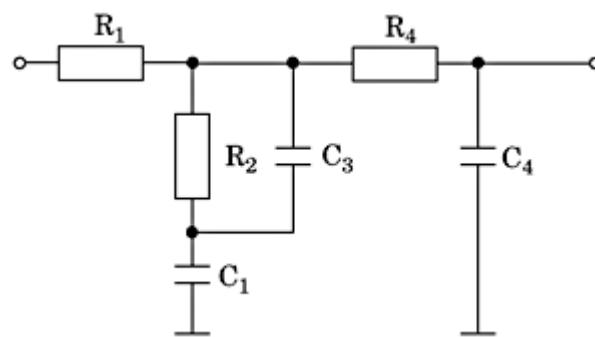
The schematic of a third-order passive lead-lag filter is shown in [Fig. 9.9](#).

Its transfer function is approximately

$$F(s) \approx \frac{1 + s(\tau_2 + \tau_3)}{1 + s(\tau_1 + \tau_2 + \tau_3) + s^2\tau_1\tau_3} \cdot \frac{1}{1 + s\tau_4} \quad (9.29)$$

with  $\tau_1 = R_1C_1$ ,  $\tau_2 = R_2C_1$ ,  $\tau_3 = R_2C_3$ , and  $\tau_4 = R_4C_4$ . This approximation is valid only if the input impedance of the filter section consisting of  $R_4$  and  $C_4$  is much larger than resistor  $R_2$ —in other words, loading of the first filter section ( $R_1$ ,  $R_2$ ,  $C_1$ ,  $C_3$ ) is negligible when  $R_4$  is at least  $5R_2$ . To place the poles and zeroes of the PLL's open-loop transfer function  $G(s)$ , it is more convenient to write this function in factorized form:

$$F(s) = \frac{1 + sT_2}{(1 + sT_1)(1 + sT_3)(1 + sT'_4)} \quad (9.30)$$



**Figure 9.9** A third-order passive RC lead-lag loop filter for voltage input.

**Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.**

For the open-loop transfer function of the PLL, we get

$$G(s) = \frac{K_0 K_d}{Ns} \cdot \frac{(1 + sT_2)}{(1 + sT_1)(1 + sT_3)(1 + sT_4)} \quad (9.31)$$

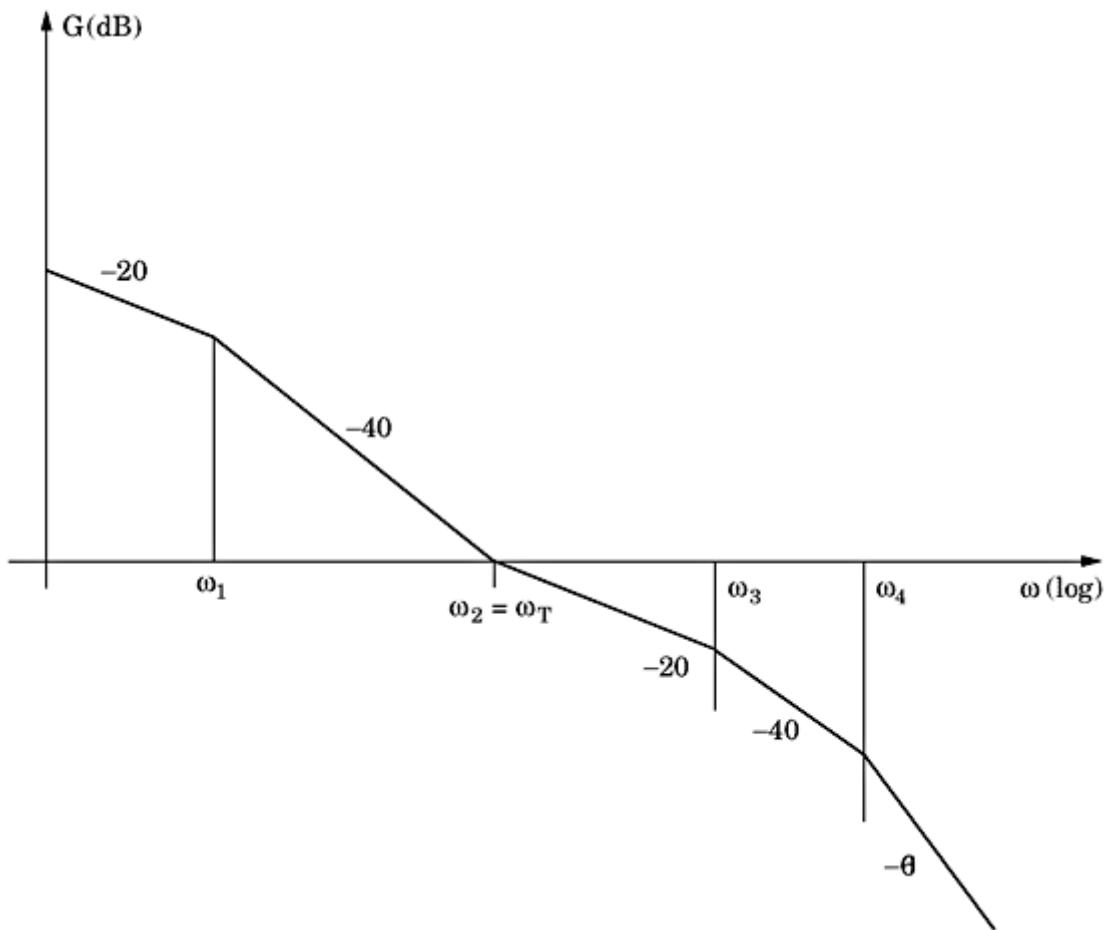
The Bode diagram of this system is plotted in Fig. 9.10.

The steps for designing the third-order passive lead-lag filter are as follows:

**Step 1** Normally, the designer knows what should be the bandwidth of the PLL, hence it is reasonable to start by choosing a value for  $\omega_{3\text{dB}}$ . A default value could be  $\omega_{3\text{dB}} = 0.05 \cdot \omega_0'$ , where  $\omega_0'$  is the down-scaled center (radian) frequency of the PLL.

**Step 2** Using the approximation in Eq. (9.4), we compute  $\omega_T = \omega_{3\text{dB}}/1.33$ . With the passive lead-lag filter, there is an upper limit for  $\omega_T$ , however, which is given by

$$\omega_{T,\text{max}} = \frac{K_0 K_d}{N} \quad (9.32)$$



**Figure 9.10** The magnitude plot of a fourth-order PLL with a passive lead-lag loop filter for voltage drive.

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

When a larger value of  $\omega_T$  is specified, this later results in a negative value for  $\tau_1$  (cf. [Eq. 9.29](#)), hence that filter would be unrealizable.

**Step 3** The corner frequencies  $\omega_2$ ,  $\omega_3$ , and  $\omega_4$  are now chosen. To get sufficient phase margin, it is recommended to set  $\omega_2 = \omega_T$  as explained in [Sec. 9.2](#). Then, for good loop stability, we recommend setting  $\omega_3 = 5\omega_2$  and  $\omega_4 = 5\omega_3$ .

**Step 4** Parameter  $T_1$  is chosen such that the open-loop gain is 1 at  $\omega = \omega_T$ . This leads to

$$T_1 = \frac{K_0 K_d}{N \omega_T^2} \quad (9.33)$$

**Step 5** The time constants  $T_2$ ,  $T_3$ , and  $T_4$  are computed from

$$T_2 = 1/\omega_2, T_3 = 1/\omega_3, \text{ and } T_4 = 1/\omega_4$$

**Step 6** Given  $T_1$  through  $T_4$ , we can calculate the filter time constants  $\tau_1$  through  $\tau_4$  now. By comparing coefficients in Eqs. [\(9.29\)](#) and [\(9.30\)](#), we get

$$\begin{aligned} \tau_1 &= T_1 + T_3 - T_2 \\ \tau_2 &= T_2 - \frac{T_1 T_3}{T_1 + T_3 - T_2} \\ \tau_3 &= T_2 - \tau_2 \\ \tau_4 &= T_4 \end{aligned} \quad (9.34)$$

**Step 7 (Optional)** There are cases where the designer modifies the values for  $\tau_1$  through  $\tau_4$  or adopts the values for  $\tau_1$  through  $\tau_4$  from an earlier design. When these parameters have been directly set or altered, it becomes desirable to know the effect on the values for  $T_1$  through  $T_4$ —in other words, on the corner frequencies  $\omega_1$  through  $\omega_4$  in the magnitude plot. When solving the Eqs. [\(9.34\)](#) for  $T_1$  through  $T_4$ , we get

(9.35)

$$T_1 = \frac{\tau_1 + \tau_2 + \tau_3 + \sqrt{(\tau_1 + \tau_2 + \tau_3)^2 - 4\tau_1\tau_3}}{2}$$

$$T_2 = \tau_2 + \tau_3$$

$$T_3 = \frac{\tau_1\tau_3}{T_1}$$

$$T_4 = \tau_4$$

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

**Step 8** Finally, the filter components ( $R, C$ ) are computed from the values for  $\tau_1$  through  $\tau_4$ . The value of  $C_1$  can be chosen arbitrarily. It should be set such that “reasonable” values for the resistors are obtained—in other words, in the region of about 1 through 100 k $\Omega$ .

### The passive lead-lag loop filter for current input

The schematic of the passive lead-lag filter for current input (charge pump) is shown in Fig. 9.11.

The open-loop transfer function of the third-order PLL then becomes

$$G(s) = \frac{K_P K_0}{s^2 C_1 N} \cdot \frac{1 + s(\tau_2 + \tau_3)}{(1 + s\tau_3)(1 + s\tau_4)} \quad (9.36)$$

with  $\tau_2 = R_2 C_1$ ,  $\tau_3 = R_2 C_3$ , and  $\tau_4 = R_4 C_4$ . This approximation is valid only if the input impedance of the filter section consisting of  $R_4$  and  $C_4$  is much larger than resistor  $R_2$ ; thus, loading the first filter section ( $R_1$ ,  $R_2$ ,  $C_1$ ,  $C_3$ ) is negligible when  $R_4$  is at least  $5R_2$ . To place the poles and zeroes of the PLL’s open-loop transfer function  $G(s)$ , it is more convenient to write this function in factorized form:

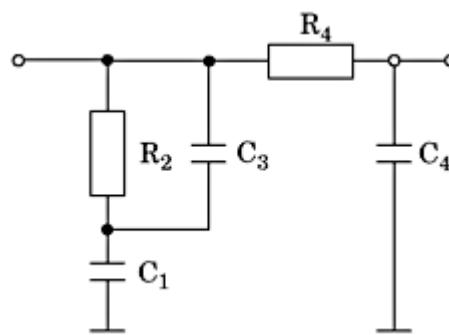
$$G(s) = \frac{K_P K_0}{s^2 C_1 N} \cdot \frac{1 + sT_2}{(1 + sT_3)(1 + sT_4)} \quad (9.37)$$

The magnitude plot of this system is shown in Fig. 9.12.

To determine the corner frequencies  $\omega_2$  and  $\omega_3$ , the transition frequency  $\omega_T$  must be known first. It has proven most convenient to proceed as described in the following steps:

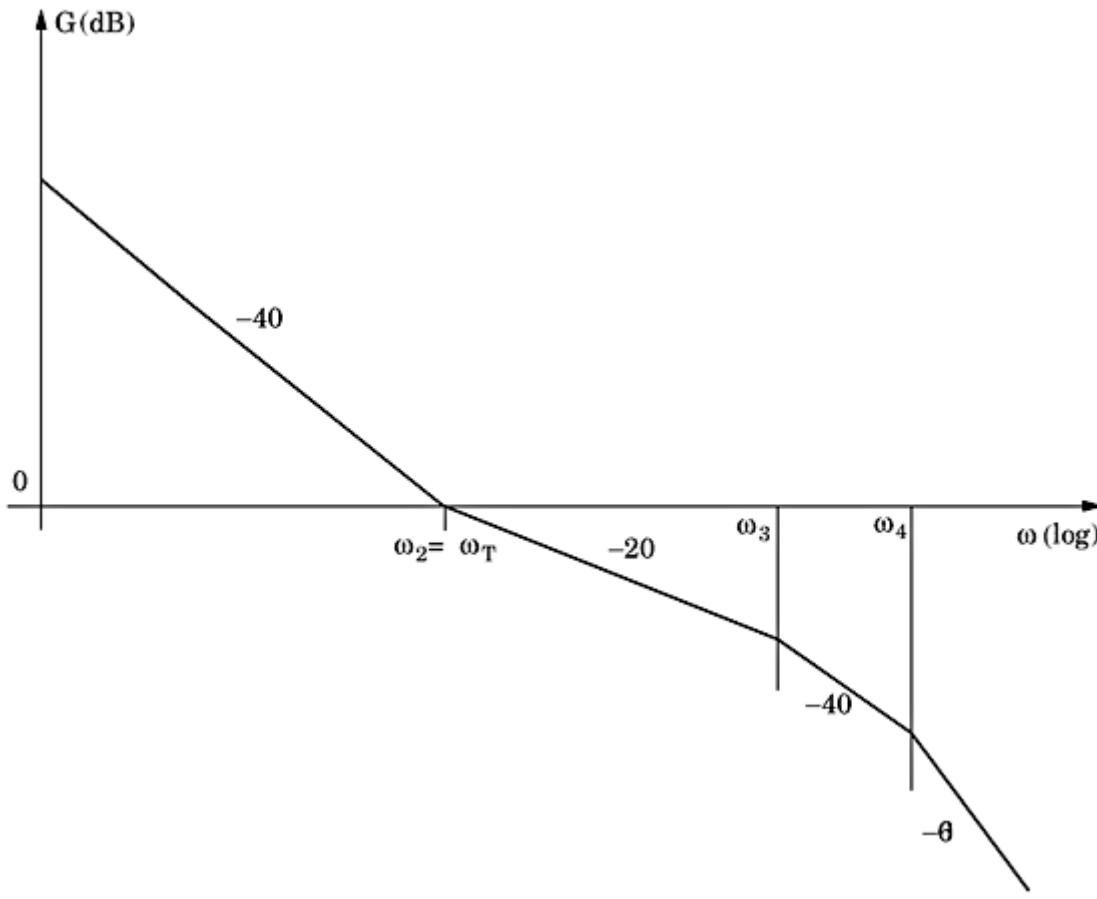
**Step 1** Normally, the designer knows what should be the bandwidth of the PLL; hence, it is reasonable to start by choosing a value for  $\omega_{3dB}$ . A default value could be  $\omega_{3dB} = 0.05 \cdot \omega_0'$ , where  $\omega_0'$  is the down-scaled center (radian) frequency of the PLL.

**Step 2** Using the approximation in Eq. (9.4), we compute  $\omega_T = \omega_{3dB}/1.33$ .



**Figure 9.11** A passive lead-lag loop filter for use with phase detectors having current output.

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.



**Figure 9.12** The magnitude plot of the open-loop gain of a fourth-order PLL using a charge pump PFD and a passive lead-lag loop filter.

**Step 3** The corner frequencies  $\omega_2$  is chosen to coincide with  $\omega_T$ , which results in a phase margin of around  $45^\circ$ . For good loop stability, we then set  $\omega_3 = 5\omega_2$  and  $\omega_4 = 5\omega_3$ , as done in the previous example ([Sec. 9.4.1](#)).

**Step 4** The capacitor  $C_1$  must be chosen such that the magnitude of the open-loop gain becomes 1 at the transition frequency. This results in

$$C_1 = \frac{K_0 K_P}{N \omega_T^2} \quad (9.38)$$

**Step 5** The time constants  $T_2$ ,  $T_3$ , and  $T_4$  are computed from

$$T_2 = 1/\omega_2, T_3 = 1/\omega_3, \text{ and } T_4 = 1/\omega_4$$

**Step 6** Given  $T_2$ ,  $T_3$ , and  $T_4$ , we can now calculate the filter time constants  $\tau_2$  and  $\tau_3$ . By comparing the coefficients in Eqs. [\(9.36\)](#) and [\(9.37\)](#), we get

$$\begin{aligned}\tau_2 &= T_2 - T_3 \\ \tau_3 &= T_3 \\ \tau_4 &= T_4\end{aligned}\tag{9.39}$$

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

**Step 7** (Optional) There are cases where the designer modifies the values for  $\tau_2$  through  $\tau_4$  or adopts the values for  $\tau_2$  through  $\tau_4$  from an earlier design. When these parameters have been directly set or altered, we certainly want to know what the effect is on the quantities  $T_2$  through  $T_4$ —in other words, on the corner frequencies  $\omega_2$  through  $\omega_4$  in the magnitude plot. When solving the Eqs. (9.39) for  $T_2$ ,  $T_3$ , and  $T_4$ , we get

$$\begin{aligned} T_2 &= \tau_1 + \tau_3 \\ T_3 &= \tau_3 \\ T_4 &= \tau_4 \end{aligned} \tag{9.40}$$

**Step 8** Finally, the filter components ( $R, C$ ) are computed from the values for  $\tau_2$  through  $\tau_4$  [cf. Eq. (9.36)].

### The active lead-lag loop filter

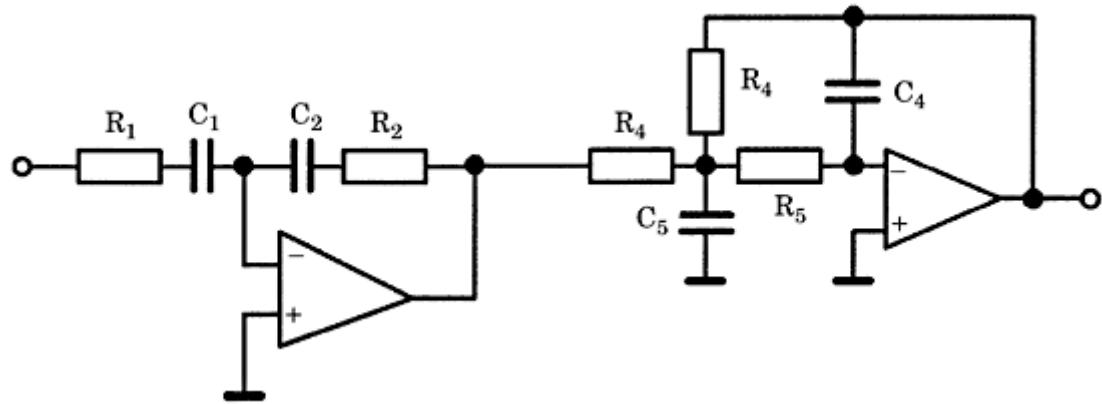
Third-order filters can be realized in many ways. One version of a third-order lead-lag filter is shown in Fig. 9.13.

The filter consists of two sections. Section 1 is a first-order lead-lag filter and is identical with the filter used to build second-order PLLs (cf. Chap. 2). Section 2 is a second-order lowpass filter. It can be realized with two real poles or with a complex-conjugate pole pair. When the poles of the second filter section are real, it is most convenient to write the filter's transfer function  $F(s)$  in the form

$$F(s) = K_a \frac{1 + s\tau_2}{1 + s\tau_1} \cdot \frac{1}{(1 + s\tau_3)(1 + s\tau_4)} \tag{9.41a}$$

with  $\tau_1 = R_1 C_1$ ,  $\tau_2 = R_2 C_2$ , and  $K_a = C_1/C_2$ . When the poles of the second filter section form a complex pair, it is more practical to use a different notation for  $F(s)$

$$F(s) = K_a \frac{1 + s\tau_2}{1 + s\tau_1} \cdot \frac{1}{1 + \frac{2s\zeta_s}{\omega_s} + \frac{s^2}{\omega_s^2}} \tag{9.41b}$$



**Figure 9.13** A third-order active lead-lag filter.

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

In the second case,  $\omega_s$  is the natural frequency and  $\zeta_s$  is the damping factor of the second-order frequency response. For the open-loop transfer function  $G(s)$  of the fourth-order PLL, we get two versions: one for the case “real poles,” and one for the case “complex poles.” For real poles,  $G(s)$  reads

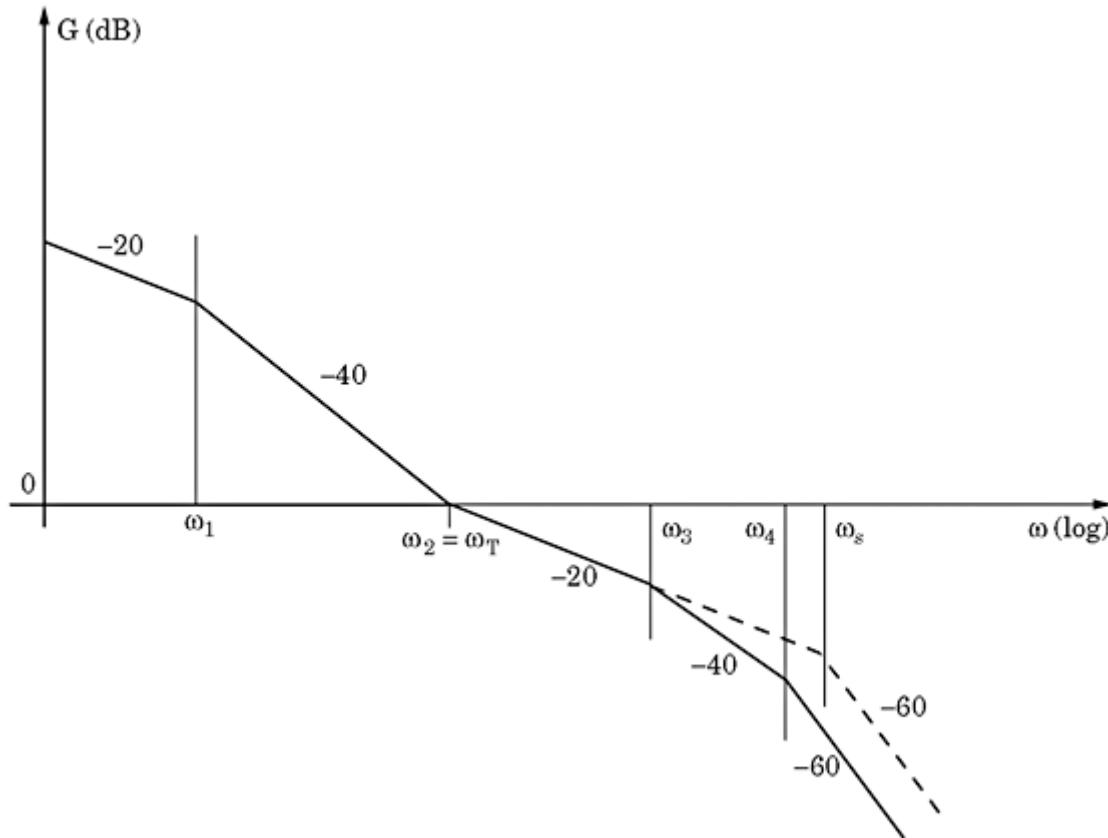
$$G(s) = \frac{K_0 K_d K_a}{Ns} \cdot \frac{1 + s\tau_2}{1 + s\tau_1} \cdot \frac{1}{(1 + s\tau_3)(1 + s\tau_4)} \quad (9.42a)$$

For complex poles,  $G(s)$  reads

$$G(s) = \frac{K_0 K_d K_a}{Ns} \cdot \frac{1 + s\tau_2}{1 + s\tau_1} \cdot \frac{1}{1 + \frac{2s\zeta_s}{\omega_s} + \frac{s^2}{\omega_s^2}} \quad (9.42b)$$

Let's have a look at the magnitude plot now (see Fig. 9.14).

The first filter section has a pole at  $s = -\omega_1$  and a zero at  $s = -\omega_2$ , and therefore provides the break points of the asymptotic magnitude plot at  $\omega_1$  and  $\omega_2$ . Suppose for the moment that the poles of the second filter section are real and



**Figure 9.14** The Bode plot for the fourth-order PLL. The solid curve represents the case where the poles of the second-order filter are real. The dashed curve shows the case

where these poles form a complex pair.

located at  $s = -\omega_3$  and  $s = -\omega_4$ . Then, that filter section creates the two corners at  $\omega_3$  and  $\omega_4$ . At  $\omega_3$  and at  $\omega_4$ , the slope of the magnitude curve increases by  $-20$  dB/decade. When the poles of the second filter section are complex, however, there is only one additional break point at  $\omega_s$ . At this point, the slope increases by  $-40$  dB/decade.

We now must provide a procedure for placing the poles and the zero of the third-order filter. It includes the following steps:

**Step 1** Normally, the designer knows what the bandwidth of the PLL should be, hence it is reasonable to start by choosing a value for  $\omega_{3dB}$ . A default value could be  $\omega_{3dB} = 0.05 \cdot \omega_0'$ , where  $\omega_0'$  is the down-scaled center (radian) frequency of the PLL.

**Step 2** Using the approximation in [Eq. \(9.4\)](#), we compute  $\omega_T = \omega_{3dB}/1.33$ . There is no mathematical restriction on the size of  $\omega_T$  here, because we have an additional parameter  $K_a$  that always can be specified such that the open-loop gain becomes 1 at the transition frequency  $\omega_T$ .

**Step 3** The corner frequency  $\omega_2$  is now chosen. To get sufficient phase margin, we set  $\omega_2 = \omega_T$ , as done previously.

**Step 4** It is recommended to set  $\omega_1 = \omega_2/10$ , thus the open-loop gain rolls off with  $-40$  dB/decade over one full decade of frequency.  $K_a$  must be chosen such that the open-loop gain becomes 1 at  $\omega_T$ . For  $K_a$ , we get

$$K_a = \frac{10N\omega_T}{K_0 K_d} \quad (9.43)$$

**Step 5** The time constants  $\tau_1$  and  $\tau_2$  are computed from  $\tau_1 = 1/\omega_1$  and  $\tau_2 = 1/\omega_2$ . Finally, the elements ( $R, C$ ) of the first filter sections are computed from  $\tau_1 = R_1 C_1$ ,  $\tau_2 = R_2 C_2$ , and  $K_a = C_1/C_2$ .  $C_1$  can be chosen arbitrarily. It should be set such that the values of the resistors are in a “reasonable” range—in other words, 1 to  $100$  k $\Omega$ .

The parameters of filter section 1 are now determined. Next, we must calculate the parameters of the second section. The procedure depends on the type of poles. When complex poles are desired, it continues with step 6; otherwise, with step 8.

**Step 6** (Complex poles) We must choose a suitable value for natural frequency  $\omega_s$  and damping factor  $\zeta_s$ . For  $\zeta_s$ , it is optimum to set  $\zeta_s = 0.707$ . Experience shows that sufficient phase margin is obtained if  $\omega_s$  is placed at  $5 \cdot \omega_T$ , hence we set  $\omega_s = 5\omega_T$ .

**Step 7** Given  $\omega_s$  and  $\zeta_s$ , we can determine the elements ( $R, C$ ) of the second filter section. Because there are more unknown elements than equations,  $C_4$  can be chosen arbitrarily. It should be set such that we obtain “reasonable” values for the resistors—in other words, in the range of 1 to  $100$  k $\Omega$ .

**Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.**

The remaining components are computed from

$$C_5 = \frac{2C_4}{\zeta_s^2} \quad (9.44)$$

$$R_4 = \sqrt{\frac{2}{\omega_s^2 C_4 C_5}}$$

$$R_5 = \frac{R_4}{2}$$

**Step 8** (Real poles) We must now choose values for the break frequencies  $\omega_3$  and  $\omega_4$  (cf. Fig. 9.14). It is recommended to set  $\omega_3 = 5\omega_T$  and  $\omega_4 = 5 \cdot \omega_3$ . We now can determine  $\tau_3 = 1/\omega_3$  and  $\tau_4 = 1/\omega_4$ .

**Step 9** To determine the elements of section 2, it is most practical to convert the filter function in Eq. (9.41a) into the filter function given in Eq. (9.41b)—thus, we would compute  $\omega_s$  and  $\zeta_s$  from the given  $\omega_3$  and  $\omega_4$ . Comparing coefficients in Eqs. (9.42a) and (9.42b) yields

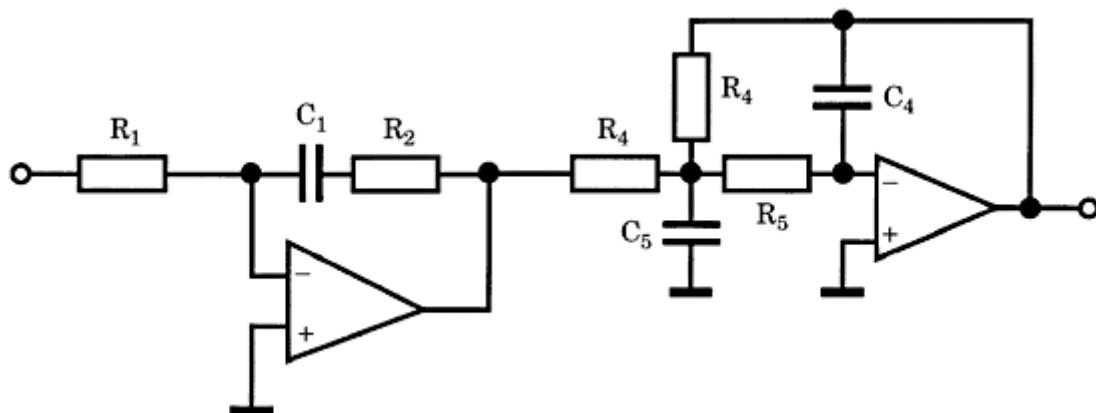
$$\omega_s = \frac{1}{\sqrt{\tau_3 \tau_4}} \quad (9.45)$$

$$\zeta_s = \frac{\tau_3 + \tau_4}{2\sqrt{\tau_3 \tau_4}}$$

Given  $\omega_s$  and  $\zeta_s$ , we now can compute the  $R$  and  $C$  values using Eqs. (9.44).

### The active PI loop filter

Third-order active PI filters can be implemented in many ways. One version is shown in Fig. 9.15.



**Figure 9.15** A third-order active PI loop filter.

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

This filter also consists of two sections. Section 1 is a first-order PI filter that has one pole at  $s = 0$  and one zero at  $s = -\omega_2$ . Section 2 has two poles. They can be either on the negative real axis in the  $s$ -plane, or they can form a complex pair. As was done previously, we will also use two forms for the filter transfer function  $F(s)$ , one for “real poles” and one for “complex poles.” For the case of real poles,  $F(s)$  is given by

$$F(s) = \frac{1 + s\tau_2}{s\tau_1} \cdot \frac{1}{(1 + s\tau_3)(1 + s\tau_4)} \quad (9.46a)$$

with  $\tau_1 = R_1 C_1$  and  $\tau_2 = R_2 C_1$ . When the poles of the second filter section form a complex pair, it is more practical to use a different notation for  $F(s)$

$$F(s) = \frac{1 + s\tau_2}{s\tau_1} \cdot \frac{1}{1 + \frac{2s\zeta_s}{\omega_s} + \frac{s^2}{\omega_s^2}} \quad (9.46b)$$

In the second case  $\omega_s$  is the natural frequency and  $\zeta_s$  the damping factor of the second-order frequency response. For the open-loop transfer function  $G(s)$  of the fourth-order PLL, we again get two versions: one for the case “real poles,” and one for the case “complex poles.” For real poles,  $G(s)$  reads

$$G(s) = \frac{K_0 K_d}{Ns} \cdot \frac{1 + s\tau_2}{s\tau_1} \cdot \frac{1}{(1 + s\tau_3)(1 + s\tau_4)} \quad (9.47a)$$

For complex poles,  $G(s)$  reads

$$G(s) = \frac{K_0 K_d}{Ns} \cdot \frac{1 + s\tau_2}{s\tau_1} \cdot \frac{1}{1 + \frac{2s\zeta_s}{\omega_s} + \frac{s^2}{\omega_s^2}} \quad (9.47b)$$

Let's have a look at the magnitude plot now (see [Fig. 9.16](#)).

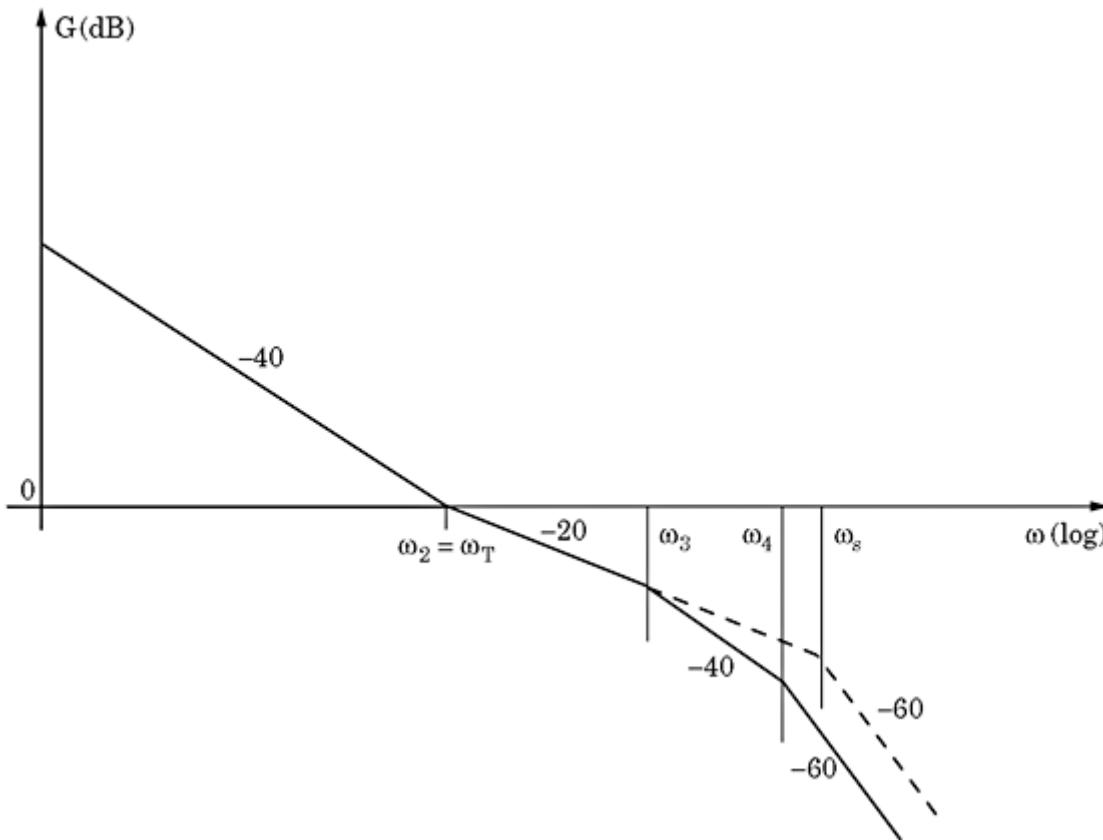
The first filter section has a pole at  $s = 0$  and a zero at  $s = -\omega_2$ , and therefore provides the breakpoint of the asymptotic magnitude plot at  $\omega_2$ . Suppose for the moment that the poles of the second filter section are real and located at  $s = -\omega_3$  and  $s = -\omega_4$ . Then that filter section creates the two corners at  $\omega_3$  and  $\omega_4$ . At  $\omega_3$  and  $\omega_4$ , the slope of the magnitude curve increases by  $-20$  dB/decade. When the poles of the second filter section are complex, however, there is only one additional breakpoint at  $\omega_s$ . At this point, the slope increases by  $-40$  dB/decade.

We now must provide a procedure for placing the poles and the zero of the third-order filter. It includes the following steps:

**Step 1** Normally, the designer knows what should be the bandwidth of the PLL, hence it is reasonable to start by choosing a value for  $\omega_{3dB}$ . A default value

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.



**Figure 9.16** The Bode plot (magnitude) of the fourth-order PLL. The solid curve represents the case where the poles of the second-order filter are real. The dashed curve shows the case where these poles form a complex pair.

could be  $\omega_{3\text{dB}} = 0.05 \cdot \omega_0'$ , where  $\omega_0'$  is the down-scaled center (radian) frequency of the PLL.

**Step 2** Using the approximation in Eq. (9.4), we compute  $\omega_T = \omega_{3\text{dB}}/1.33$ . There is no mathematical restriction on the size of  $\omega_T$  here, because time constant  $\omega_T$  can be set such that the open-loop gain becomes 1 at the transition frequency  $\omega_T$ .

**Step 3** The corner frequency  $\omega_2$  is now chosen. To get sufficient phase margin, we set  $\omega_2 = \omega_T$ .

**Step 4** Parameter  $\tau_1$  is chosen such that the open-loop gain is 1 at  $\omega = \omega_T$ . This leads to

$$\tau_1 = \frac{K_0 K_d}{N \omega_T^2} \quad (9.48)$$

**Step 5** Time constant  $\tau_2$  is computed from  $\tau_2 = 1/\omega_T$ . Now the elements ( $R, C$ ) of the first filter section are computed from  $\tau_1 = R_1 C_1$  and  $\tau_2 = R_2 C_1$ .  $C_1$  can be chosen arbitrarily. It

should be set such that the values for  $R_1$  and  $R_2$  become “reasonable”—that is, in the range of 1 to 100 k $\Omega$ .

The parameters of filter section 1 are now determined. Next, we must calculate the parameters of the second section. The procedure depends on the type of poles. When complex poles are desired, it continues with step 6; otherwise, with step 8.

**Step 6** (Complex poles) We must choose a suitable value for natural frequency  $\omega_s$  and damping factor  $\zeta_s$ . For  $\zeta_s$ , it is optimum to set  $\zeta_s = 0.707$ . Experience shows that sufficient phase margin is obtained if  $\omega_s$  is placed at  $5 \cdot \omega_T$ , hence we set  $\omega = 5\omega_T$ .

**Step 7** Given  $\omega_s$  and  $\zeta_s$ , we can determine the elements  $(R, C)$  of the second filter section. Because there are more unknown elements than equations,  $C_4$  can be chosen arbitrarily. It should be set such that we obtain “reasonable” values for the resistors—in other words, in the range of 1 to 100 kΩ.

The remaining components are computed from

$$C_5 = \frac{2C_4}{\zeta_s^2} \quad (9.49)$$

$$R_4 = \sqrt{\frac{2}{\omega_s^2 C_4 C_5}}$$

$$R_5 = \frac{R_4}{2}$$

**Step 8** (Real poles) We must choose values for the break frequencies  $\omega_3$  and  $\omega_4$  now (cf. Fig. 9.16). It is recommended to set  $\omega_3 = 5\omega_T$  and  $\omega_4 = 5\omega_3$ . We now can determine  $\tau_3 = 1/\omega_3$  and  $\tau_4 = 1/\omega_4$ .

**Step 9** To determine the elements of section 2, it is most practical to convert the filter function in Eq. (9.46a) into the filter function given in Eq. (9.46b)—thus, we would compute  $\omega_s$  and  $\zeta_s$  from the given  $\omega_3$  and  $\omega_4$ . Comparing coefficients in Eqs. (9.46a) and (9.46b) yields

$$\omega_s = \frac{1}{\sqrt{\tau_3 \tau_4}} \quad (9.50)$$

$$\zeta_s = \frac{\tau_3 + \tau_4}{2\sqrt{\tau_3 \tau_4}}$$

Given  $\omega_s$  and  $\zeta_s$ , we now can compute the  $R$  and  $C$  values using Eqs. (9.49).

## Designing Fifth-Order PLLs

When designing a fifth-order PLL, a fourth-order loop filter must be selected. As in the previous section, four types of loop filters will be analyzed: the passive lead-lag filter with voltage drive, the passive lead-lag filter with current drive, the active lead-lag filter, and the active PI filter.

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

## Passive lead-lag loop filter for voltage input

A fourth-order passive lead-lag filter is shown in Fig. 9.17.

Its transfer function is approximately

$$F(s) \approx \frac{1 + s(\tau_2 + \tau_3)}{1 + s(\tau_1 + \tau_2 + \tau_3) + s^2\tau_1\tau_3} \cdot \frac{1}{1 + s\tau_4} \cdot \frac{1}{1 + s\tau_5} \quad (9.51)$$

with  $\tau_1 = R_1 C_1$ ,  $\tau_2 = R_2 C_1$ ,  $\tau_3 = R_2 C_3$ ,  $\tau_4 = R_4 C_4$ , and  $\tau_5 = R_5 C_5$ . This approximation is valid only if the input impedance of the filter section consisting of  $R_4$  and  $C_4$  is much larger than resistor  $R_2$ —in other words, loading of the first filter section ( $R_1$ ,  $R_2$ ,  $C_1$ ,  $C_3$ ) is negligible when  $R_4$  is at least  $5R_2$ , and if resistor  $R_5$  is at least five times larger than  $R_4$ . To place the poles and zeroes of the PLL's open-loop transfer function  $G(s)$ , it is more convenient to write this function in factorized form:

$$F(s) = \frac{1 + sT_2}{(1 + sT_1)(1 + sT_3)(1 + sT_4)(1 + sT_5)} \quad (9.52)$$

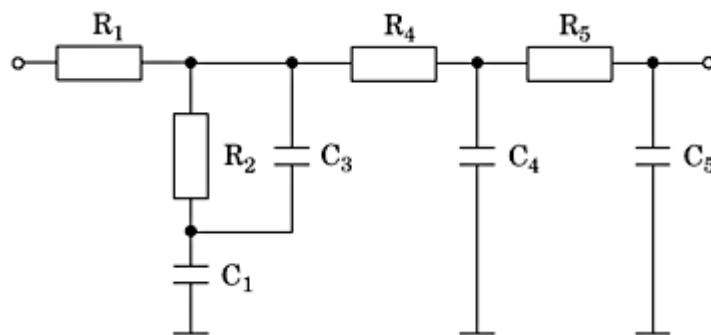
For the open-loop transfer function of the PLL, we get

$$G(s) = \frac{K_0 K_d}{Ns} \cdot \frac{(1 + sT_2)}{(1 + sT_1)(1 + sT_3)(1 + sT_4)(1 + sT_5)} \quad (9.53)$$

The magnitude plot for the fifth-order PLL is given in Fig. 9.18.

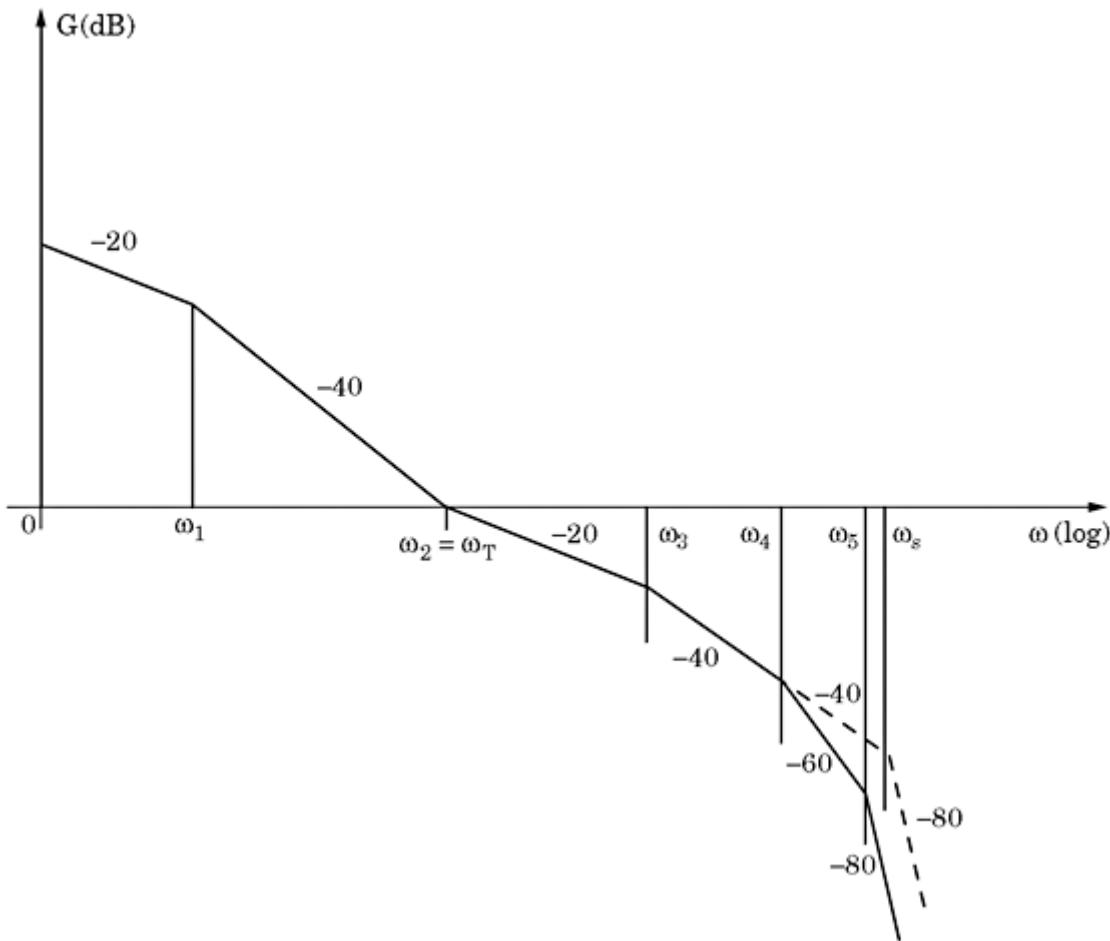
The steps for designing the fourth-order passive lead-lag filter are as follows:

**Step 1** Normally, the designer knows what the bandwidth of the PLL should be, hence it is reasonable to start by choosing a value for  $\omega_{3dB}$ . A default value could be  $\omega_{3dB} = 0.05 \cdot \omega_0'$ , where  $\omega_0'$  is the down-scaled center (radian) frequency of the PLL.



**Figure 9.17** A fourth-order passive lead-lag loop filter with voltage drive.

**Any use is subject to the Terms of Use as given at the website.**



**Figure 9.18** The magnitude plot of fifth-order PLL built with a passive lead-lag loop filter for voltage drive.

**Step 2** Using the approximation in Eq. (9.4), we compute  $\omega_T = \omega_{3\text{dB}}/1.33$ . With the passive lead-lag filter, there is an upper limit for  $\omega_T$ , however, which is given by

$$\omega_{T,\max} = \frac{K_0 K_d}{N} \quad (9.54)$$

When a larger value of  $\omega_T$  is specified, this later results in a negative value for  $\tau_1$  (cf. Eq. 9.51), hence that filter would be unrealizable.

**Step 3** The corner frequencies  $\omega_2$ ,  $\omega_3$ ,  $\omega_4$ , and  $\omega_5$  are now chosen. To get sufficient phase margin, it is recommended to set  $\omega_2 = \omega_T$ , as explained in Sec. 9.2. Then for good loop stability we recommend setting  $\omega_3 = 5\omega_2$ ,  $\omega_4 = 5\omega_3$ , and  $\omega_5 = 5\omega_4$ .

**Step 4** Parameter  $T_1$  is chosen such that the open-loop gain is 1 at  $\omega = \omega_T$ . This leads to

$$T_1 = \frac{K_0 K_d}{N \omega_T^2} \quad (9.55)$$

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

**Step 5** The time constants  $T_2$ ,  $T_3$ ,  $T_4$ , and  $T_5$  are computed from

$$T_2 = 1/\omega_2, T_3 = 1/\omega_3, T_4 = 1/\omega_4, \text{ and } T_5 = 1/\omega_5.$$

**Step 6** Given  $T_1$  through  $T_5$ , we can calculate the filter time constants  $\tau_1$  through  $\tau_5$  now. By comparing coefficients in Eqs. (9.51) and (9.52), we get

$$\tau_1 = T_1 + T_3 - T_2 \quad (9.56)$$

$$\tau_2 = T_2 - \frac{T_1 T_3}{T_1 + T_3 - T_2}$$

$$\tau_3 = T_2 - \tau_2$$

$$\tau_4 = T_4$$

$$\tau_5 = T_5$$

**Step 7** (Optional) There are cases where the designer modifies the values for  $\tau_1$  through  $\tau_5$  or adopts the values for  $\tau_1$  through  $\tau_5$  from an earlier design. When these parameters have been directly set or altered, it becomes desirable to know the effect on the values for  $T_1$  through  $T_5$ —in other words, on the corner frequencies  $\omega_1$  through  $\omega_5$  in the magnitude plot. When solving the Eqs. (9.56) for  $T_1$  through  $T_5$ , we get

$$T_1 = \frac{\tau_1 + \tau_2 + \tau_3 + \sqrt{(\tau_1 + \tau_2 + \tau_3)^2 - 4\tau_1\tau_3}}{2} \quad (9.57)$$

$$T_2 = \tau_2 + \tau_3$$

$$T_3 = \frac{\tau_1 \tau_3}{T_1}$$

$$T_4 = \tau_4$$

$$T_5 = \tau_5$$

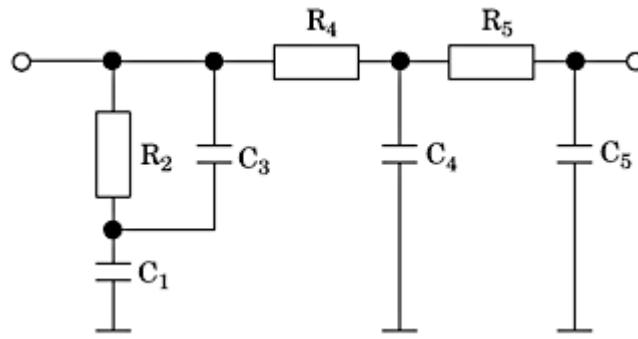
**Step 8** Finally, the filter components ( $R, C$ ) are computed from the values for  $\tau_1$  through  $\tau_5$  [cf. Eq. (9.51)]. The value of  $C_1$  can be chosen arbitrarily. It should be set such that “reasonable” values for the resistors are obtained—in other words, in the region of about 1 to 100 k $\Omega$ .

### The passive lead-lag loop filter for current input

The schematic of the fourth-order passive lead-lag loop filter for current input (charge pump) is shown in [Fig. 9.19](#).

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.



**Figure 9.19** A passive lead-lag loop filter for use with phase detectors having current output.

The open-loop transfer function of the fifth-order PLL becomes approximately

$$G(s) \approx \frac{K_p K_0}{s^2 C_1 N} \cdot \frac{1 + s(\tau_2 + \tau_3)}{(1 + s\tau_3)(1 + s\tau_4)(1 + s\tau_5)} \quad (9.58)$$

with  $\tau_2 = R_2 C_1$ ,  $\tau_3 = R_2 C_3$ ,  $\tau_4 = R_4 C_4$ , and  $\tau_5 = R_5 C_5$ . This approximation is valid only if the input impedance of the filter section consisting of  $R_4$  and  $C_4$  is much larger than resistor  $R_2$ —in other words, loading of the first filter section ( $R_1$ ,  $R_2$ ,  $C_1$ ,  $C_3$ ) is negligible when  $R_4$  is at least  $5R_2$ , and if resistor  $R_5$  is at least five times larger than  $R_4$ .

To place the poles and zeroes of the PLL's open-loop transfer function  $G(s)$ , it is more convenient to write this function in factorized form:

$$G(s) = \frac{K_p K_0}{s^2 C_1 N} \cdot \frac{1 + sT_2}{(1 + sT_3)(1 + sT_4)(1 + sT_5)} \quad (9.59)$$

The magnitude plot of this system is shown in [Fig. 9.20](#).

To determine the corner frequencies  $\omega_2$  through  $\omega_5$ , the transition frequency  $\omega_T$  must be known first. It has proven most convenient to proceed as described in the following steps:

**Step 1** Normally, the designer knows what the bandwidth of the PLL should be, hence it is reasonable to start by choosing a value for  $\omega_{3dB}$ . A default value could be  $\omega_{3dB} = 0.05 \cdot \omega_0'$ , where  $\omega_0'$  is the down-scaled center (radian) frequency of the PLL.

**Step 2** Using the approximation in [Eq. \(9.4\)](#), we compute  $\omega_T = \omega_{3dB}/1.33$ .

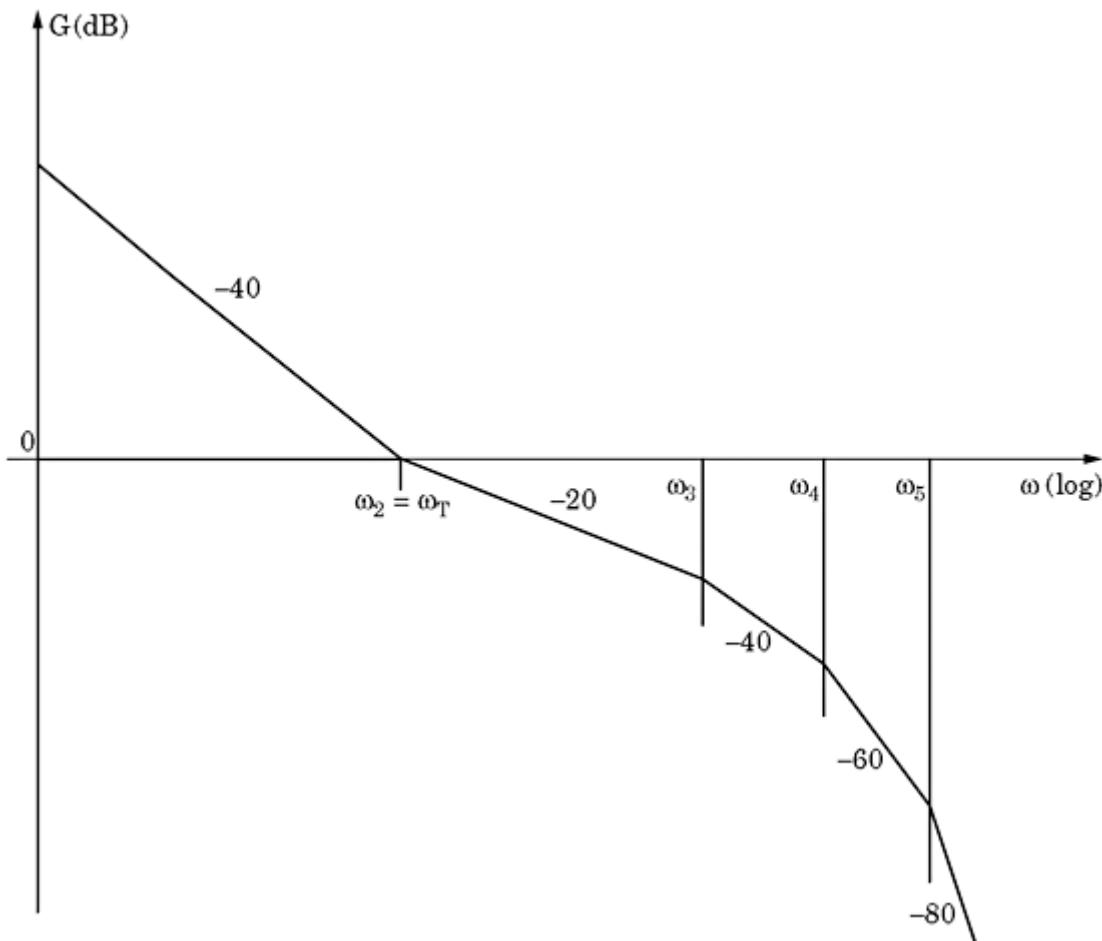
**Step 3** The corner frequencies  $\omega_2$  is chosen to coincide with  $\omega_T$ , which results in a phase margin of around  $45^\circ$ . For good loop stability, we then set  $\omega_3 = 5\omega_2$ ,  $\omega_4 = 5\omega_3$ , and  $\omega_5 = 5\omega_4$ , as done in the previous example ([Sec. 9.5.1](#)).

**Step 4** The capacitor  $C_1$  must be chosen such that the magnitude of the open-loop gain becomes 1 at the transition frequency. This results in

$$C_1 = \frac{K_0 K_P}{N \omega_T^2} \quad (9.60)$$

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.



**Figure 9.20** The magnitude plot of open-loop gain of a fifth-order PLL using a charge pump PFD and a passive lead-lag loop filter.

**Step 5** The time constants  $T_2$ ,  $T_3$ ,  $T_4$ , and  $T_5$  are computed from

$$T_2 = 1/\omega_2, T_3 = 1/\omega_3, T_4 = 1/\omega_4, \text{ and } T_5 = 1/\omega_5$$

**Step 6** Given  $T_2$ ,  $T_3$ ,  $T_4$ , and  $T_5$ , we can now calculate the filter time constants  $\tau_2$  through  $\tau_5$ . By comparing coefficients in Eqs. (9.58) and (9.59), we get

$$\begin{aligned} \tau_2 &= T_2 - T_3 \\ \tau_3 &= T_3 \\ \tau_4 &= T_4 \\ \tau_5 &= T_5 \end{aligned} \tag{9.61}$$

**Step 7** (Optional) There are cases where the designer modifies the values for  $\tau_2$  through  $\tau_5$

or adopts the values for  $\tau_2$  through  $\tau_5$  from an earlier design. When these parameters have been directly set or altered, we certainly want to know what the effect is on the quantities  $T_2$  through  $T_5$ —in other words, on the corner

frequencies  $\omega_2$  through  $\omega_5$  in the magnitude plot. When solving the Eqs. (9.61) for  $T_2$ ,  $T_3$ ,  $T_4$ , and  $T_5$ , we get

$$\begin{aligned} T_2 &= \tau_2 - \tau_3 \\ T_3 &= \tau_3 \\ T_4 &= \tau_4 \\ T_5 &= \tau_5 \end{aligned} \tag{9.62}$$

**Step 8** Finally, the filter components ( $R, C$ ) are computed from the values for  $\tau_2$  through  $\tau_5$  [cf. Eq. (9.58)].

### Active lead-lag loop filter

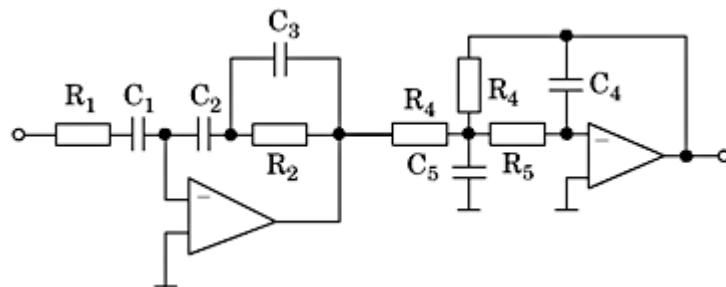
Fourth-order filters can be realized in many ways. One version of a fourth-order lead-lag filter is shown in Fig. 9.21.

The filter consists of two sections. Section 1 is a second-order lead-lag filter. It has two poles on the real axis in the  $s$ -plane and one zero. The poles are located at  $s = -\omega_1$  and  $s = -\omega_3$ , while the zero is at  $s = -\omega_2$ . Section 2 is a second-order lowpass filter. It can be realized with two real poles or with a complex-conjugate pole pair. When the poles of the second filter section are real, it is most convenient to write the filter's transfer function  $F(s)$  in the form

$$F(s) = K_a \frac{1 + s(\tau_2 + \tau_3)}{(1 + s\tau_1)(1 + s\tau_3)} \cdot \frac{1}{(1 + s\tau_4)(1 + s\tau_5)} \tag{9.63a}$$

with  $\tau_1 = R_1 C_1$ ,  $\tau_2 = R_2 C_2$ ,  $\tau_3 = R_2 C_3$ , and  $K_a = C_1/C_2$ . When the poles of the second filter section form a complex pair, it is more practical to use a different notation for  $F(s)$  such as

$$F(s) = K_a \frac{1 + s(\tau_2 + \tau_3)}{(1 + s\tau_1)(1 + s\tau_3)} \cdot \frac{1}{1 + \frac{2s\zeta_s}{\omega_s} + \frac{s^2}{\omega_s^2}} \tag{9.63b}$$



**Figure 9.21** A fourth-order active lead-lag filter.

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

In the second case  $\omega_s$  is the natural frequency and  $\zeta_s$  is the damping factor of the second-order frequency response. Because we want to determine break frequencies (corners) in the asymptotic magnitude plot, it is more convenient to write Eqs. (9.63a) and (9.63b) in standardized form. For the case “real poles,” we get

$$F(s) = K_a \frac{1 + sT_2}{(1 + sT_1)(1 + sT_3)} \cdot \frac{1}{(1 + s\tau_4)(1 + s\tau_5)} \quad (9.64a)$$

and for the case “complex poles,” we have

$$F(s) = K_a \frac{1 + sT_2}{(1 + sT_1)(1 + sT_3)} \cdot \frac{1}{1 + \frac{2s\zeta_s}{\omega_s} + \frac{s^2}{\omega_s^2}} \quad (9.64b)$$

For the open-loop transfer function  $G(s)$  of the fourth-order PLL, we get two versions, one for “real poles” and one for “complex poles.” For real poles,  $G(s)$  reads

$$G(s) = \frac{K_0 K_d K_a}{Ns} \cdot \frac{1 + sT_2}{(1 + sT_1)(1 + sT_3)} \cdot \frac{1}{(1 + s\tau_4)(1 + s\tau_5)} \quad (9.65a)$$

For complex poles,  $G(s)$  reads

$$G(s) = \frac{K_0 K_d K_a}{Ns} \cdot \frac{1 + sT_2}{(1 + sT_1)(1 + sT_3)} \cdot \frac{1}{1 + \frac{2s\zeta_s}{\omega_s} + \frac{s^2}{\omega_s^2}} \quad (9.65b)$$

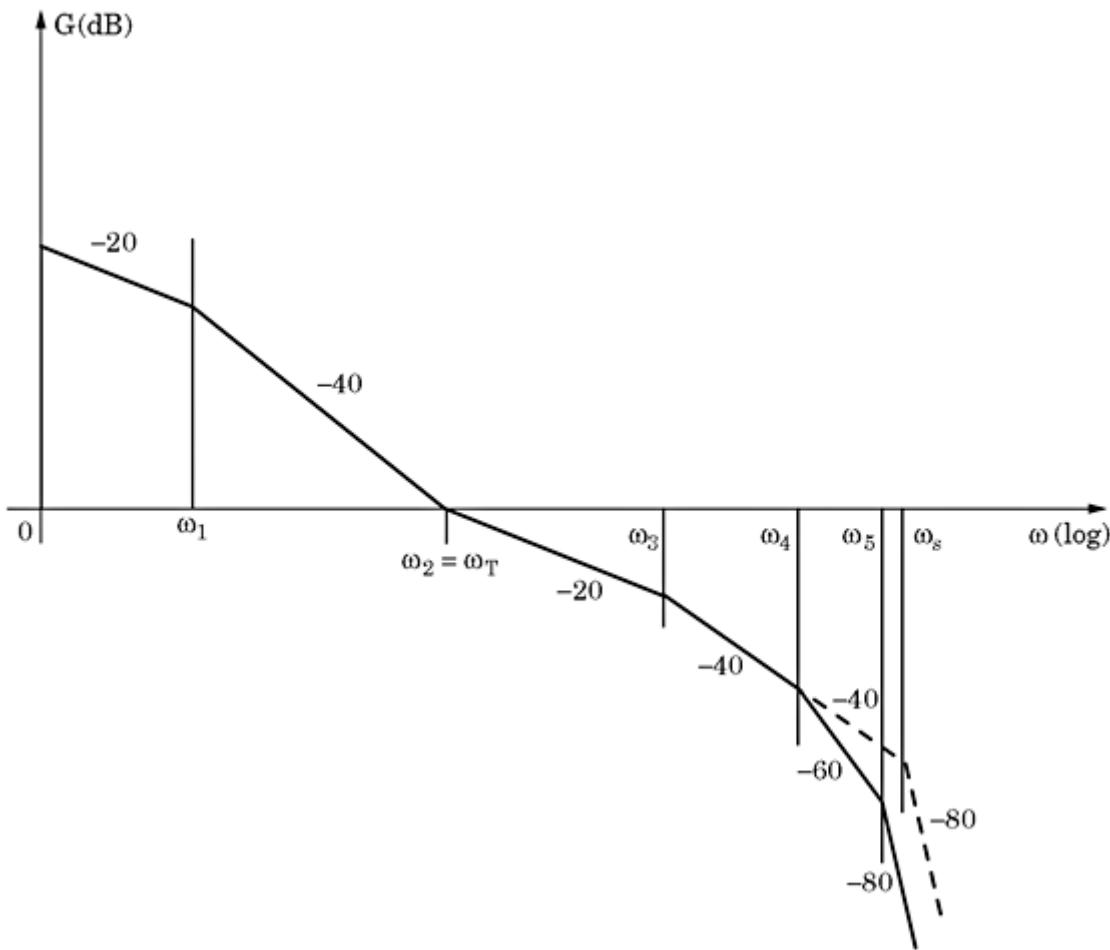
Let’s have a look at the magnitude plot now (see Fig. 9.22).

The first filter section has poles at  $s = -\omega_1$  and at  $s = -\omega_3$ , and a zero at  $s = -\omega_2$ . It therefore provides the break points of the asymptotic magnitude plot at  $\omega_1$ ,  $\omega_2$ , and  $\omega_3$ . Suppose for the moment that the poles of the second filter section are real and located at  $s = -\omega_4$  and  $s = -\omega_5$ . Then, that filter section creates the two corners at  $\omega_4$  and  $\omega_5$ . At  $\omega_4$  and  $\omega_5$ , the slope of the magnitude curve increases by  $-20$  dB/decade. When the poles of the second filter section are complex, however, there is only one additional breakpoint at  $\omega_s$ . At this point, the slope increases by  $-40$  dB/decade.

We now must provide a procedure for placing the poles and the zero of the fourth-order filter. It includes the following steps:

**Step 1** Normally, the designer knows what the bandwidth of the PLL should be, hence it is reasonable to start by choosing a value for  $\omega_{3dB}$ . A default value could be  $\omega_{3dB} = 0.05 \cdot \omega_0'$ , where  $\omega_0'$  is the down-scaled center (radian) frequency of the PLL.

**Any use is subject to the Terms of Use as given at the website.**



**Figure 9.22** The Bode plot for the fifth-order PLL. The solid curve represents the case where the poles of the second filter section are real. The dashed curve shows the case where these poles form a complex pair.

**Step 2** Using the approximation in Eq. (9.4), we compute  $\omega_T = \omega_{3\text{dB}}/1.33$ . There is no mathematical restriction on the size of  $\omega_T$ , because we have an additional parameter  $K_a$  that always can be specified such that the open-loop gain becomes 1 at the transition frequency  $\omega_T$ .

**Step 3** The corner frequencies  $\omega_2$  and  $\omega_3$  are now chosen. To get sufficient phase margin, it is recommended to set  $\omega_2 = \omega_T$  and  $\omega_3 = 5\omega_2$ .

**Step 4** It is recommended to set  $\omega_1 = \omega_2/10$ , thus the open-loop gain rolls off with  $-40 \text{ dB/decade}$  over one full decade of frequency.  $K_a$  must be chosen such that the open-loop gain just becomes 1 at  $\omega_T$ . For  $K_a$ , we get

$$K_a = \frac{10N\omega_T}{K_0 K_d} \quad (9.66)$$

**Step 5** The time constants  $T_1$ ,  $T_2$ , and  $T_3$  are computed from  $T_1 = 1/\omega_1$ ,  $T_2 = 1/\omega_2$ , and  $T_3 = 1/\omega_3$ .

**Step 6** To realize section 1 of the filter, we must also know the time constants  $\tau_1$ ,  $\tau_2$ , and  $\tau_3$ . They are computed from

$$\begin{aligned}\tau_1 &= T_1 \\ \tau_2 &= T_2 - T_3 \\ \tau_3 &= T_3\end{aligned}\tag{9.67}$$

Finally, the elements ( $R$ ,  $C$ ) of the first filter section are computed from  $\tau_1 = R_1 C_1$ ,  $\tau_2 = R_2 C_2$ ,  $\tau_3 = R_3 C_3$ , and  $K_a = C_1/C_2$ .  $C_1$  can be chosen arbitrarily. It should be set such that the resistor values are in a “reasonable” range—say, 1 to 100 kΩ.

**Step 7 (Optional)** Sometimes the designer alters the values  $\tau_1$  through  $\tau_3$  and wants to know what the effect is on the quantities  $T_1$  through  $T_3$ . The variables  $T_1$  through  $T_3$  are then computed from  $\tau_1$  through  $\tau_3$  as follows:

$$\begin{aligned}T_1 &= \tau_1 \\ T_2 &= \tau_2 + \tau_3 \\ T_3 &= \tau_3\end{aligned}\tag{9.68}$$

The parameters of filter section 1 are determined now. Next, we must calculate the parameters of the second section. The procedure depends on the type of poles. When complex poles are desired, it continues with step 8, otherwise with step 10.

**Step 8 (Complex poles)** We must choose a suitable value for natural frequency  $\omega_s$  and damping factor  $\zeta_s$ . For  $\zeta_s$ , it is optimum to set  $\zeta_s = 0.707$ . Experience shows that sufficient phase margin is obtained if  $\omega_s$  is placed at  $5 \cdot \omega_T$ , hence we set  $\omega_s = 5\omega_3$ .

**Step 9** Given  $\omega_s$  and  $\zeta_s$ , we can determine the elements ( $R, C$ ) of the second filter section. Because there are more unknown elements than equations,  $C_4$  can be chosen arbitrarily. It should be set such that we obtain “reasonable” values for the resistors—in other words, in the range of 1 to 100 kΩ.

The remaining components are computed from

$$(9.69)$$

$$C_5 = \frac{2C_4}{\zeta_s^2}$$

$$R_4 = \sqrt{\frac{2}{\omega_s^2 C_4 C_5}}$$

$$R_5 = \frac{R_4}{2}$$

**Step 10** (Real poles) We must now choose values for the break frequencies  $\omega_4$  and  $\omega_5$  (cf. Fig. 9.22). It is recommended to set  $\omega_4 = 5 \cdot \omega_3$  and  $\omega_5 = 5 \cdot \omega_4$ . We now can determine  $\tau_4 = 1/\omega_4$  and  $\tau_5 = 1/\omega_5$ .

**Step 11** To determine the elements of section 2, it is most practical to convert the filter function in Eq. (9.64a) into the filter function given in Eq. (9.64b)—thus, we would compute  $\omega_s$  and  $\zeta_s$  from the given  $\omega_4$  and  $\omega_5$ . Comparing coefficients in Eqs. (9.64a) and (9.65b) yields

$$\omega_s = \frac{1}{\sqrt{\tau_4 \tau_5}} \quad (9.70)$$

$$\zeta_s = \frac{\tau_4 + \tau_5}{2\sqrt{\tau_4 \tau_5}}$$

Given  $\omega_s$  and  $\zeta_s$ , we now can compute the  $R$  and  $C$  values using Eqs. (9.69).

### Active PI loop filter

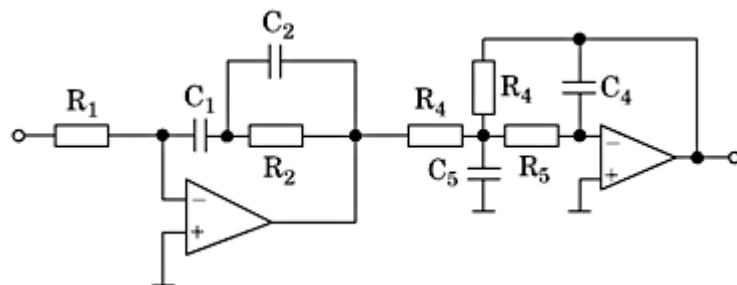
Fourth-order active PI filters can be implemented in many ways. One version is shown in Fig. 9.23.

The filter consists of two sections. Section 1 is a second-order PI filter. It has two poles on the real axis in the  $s$ -plane and one zero. The poles are located at  $s = 0$  and  $s = -\omega_3$ ; the zero is at  $s = -\omega_2$ . Section 2 is a second-order lowpass filter. It can be realized with two real poles or with a complex-conjugate pole pair. When the poles of the second filter section are real, it is most convenient to write the filter's transfer function  $F(s)$  in the form

$$F(s) = \frac{1 + s(\tau_2 + \tau_3)}{s\tau_1(1 + s\tau_3)} \cdot \frac{1}{(1 + s\tau_4)(1 + s\tau_5)} \quad (9.71a)$$

with  $\tau_1 = R_1 C_1$ ,  $\tau_2 = R_2 C_1$ , and  $\tau_3 = R_2 C_2$ . When the poles of the second filter section form a complex pair, it is more practical to use a different notation for  $F(s)$

$$F(s) = \frac{1 + s(\tau_2 + \tau_3)}{s\tau_1(1 + s\tau_3)} \cdot \frac{1}{1 + \frac{2s\zeta_s}{\omega_s} + \frac{s^2}{\omega_s^2}} \quad (9.71b)$$



**Figure 9.23** A fourth-order active PI loop filter.

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

In the second case,  $\omega_s$  is the natural frequency and  $\zeta_s$  is the damping factor of the second-order frequency response. Because we want to determine break frequencies (corners) in the asymptotic magnitude plot, it is more convenient to write Eqs. (9.71a) and (9.71b) in standardized form. For the case “real poles,” we get

$$F(s) = \frac{1 + sT_2}{sT_1(1 + sT_3)} \cdot \frac{1}{(1 + s\tau_4)(1 + s\tau_5)} \quad (9.72a)$$

and for the case “complex poles,” we have

$$F(s) = \frac{1 + sT_2}{sT_1(1 + sT_3)} \cdot \frac{1}{1 + \frac{2s\zeta_s}{\omega_s} + \frac{s^2}{\omega_s^2}} \quad (9.72b)$$

For the open-loop transfer function  $G(s)$  of the fourth-order PLL, we get two versions, one for the case “real poles” and one for the case “complex poles.” For real poles,  $G(s)$  reads

$$G(s) = \frac{K_0 K_d}{Ns} \cdot \frac{1 + sT_2}{sT_1(1 + sT_3)} \cdot \frac{1}{(1 + s\tau_4)(1 + s\tau_5)} \quad (9.73a)$$

For complex poles,  $G(s)$  reads

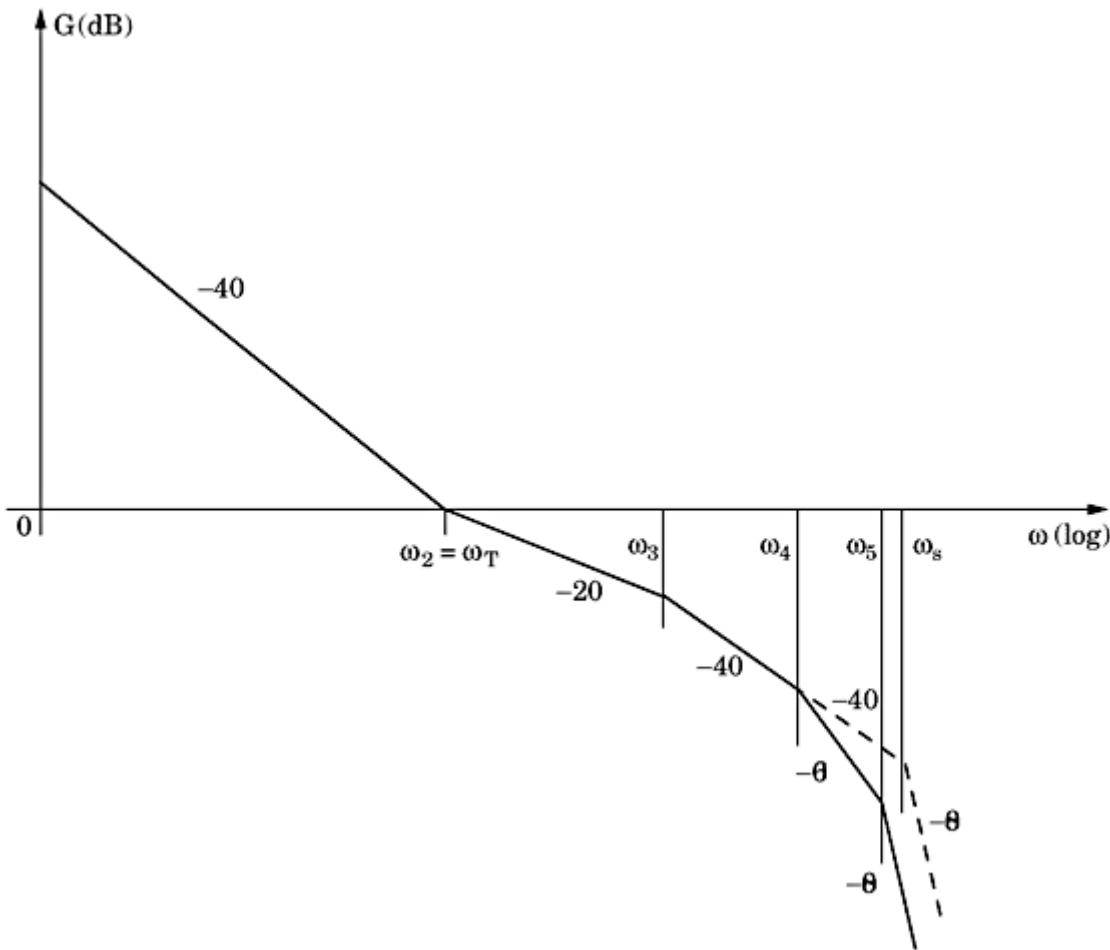
$$G(s) = \frac{K_0 K_d}{Ns} \cdot \frac{1 + sT_2}{sT_1(1 + sT_3)} \cdot \frac{1}{1 + \frac{2s\zeta_s}{\omega_s} + \frac{s^2}{\omega_s^2}} \quad (9.73b)$$

Let’s have a look at the magnitude plot now (see Fig. 9.24).

The first filter section has poles at  $s = 0$  and  $s = -\omega_3$ , and a zero at  $s = -\omega_2$ . It provides the breakpoints of the asymptotic magnitude plot at  $\omega_2$  and  $\omega_3$ . Suppose for the moment that the poles of the second filter section are real and located at  $s = -\omega_4$  and  $s = -\omega_5$ . Then that filter section creates the two corners at  $\omega_4$  and  $\omega_5$ . At  $\omega_4$  and at  $\omega_5$ , the slope of the magnitude curve increases by  $-20$  dB/decade. When the poles of the second filter section are complex, however, there is only one additional breakpoint at  $\omega_s$ . At this point, the slope increases by  $-40$  dB/decade.

We now must provide a procedure for placing the poles and the zero of the third-order filter. It includes the following steps:

**Step 1** Normally, the designer knows what the bandwidth of the PLL should be, hence it is reasonable to start by choosing a value for  $\omega_{3dB}$ . A default value



**Figure 9.24** The Bode plot (magnitude) of fifth-order PLL. The solid curve represents the case where the poles of the second filter section are real. The dashed curve shows the case where these poles form a complex pair.

could be  $\omega_{3\text{dB}} = 0.05 \cdot \omega_0'$ , where  $\omega_0'$  is the down-scaled center (radian) frequency of the PLL.

**Step 2** Using the approximation in Eq. (9.4), we compute  $\omega_T = \omega_{3\text{dB}}/1.33$ . There is no mathematical restriction on the size of  $\omega_T$  here, because time constant  $T_1$  can be set such that the open-loop gain becomes 1 at the transition frequency  $\omega_T$ .

**Step 3** The corner frequencies  $\omega_2$  and  $\omega_3$  are now chosen. To get a sufficient phase margin, it is recommended to set  $\omega_2 = \omega_T$  and  $\omega_3 = 5\omega_2$ .

**Step 4** Parameter  $T_1$  is chosen such that the open-loop gain is 1 at  $\omega = \omega_T$ . This leads to

$$T_1 = \frac{K_0 K_d}{N \omega_T^2} \quad (9.74)$$

**Step 5** Time constants  $T_2$  and  $T_3$  are computed from  $T_2 = 1/\omega_2$  and  $T_3 = 1/\omega_3$ .

**Step 6** To realize section 1 of the loop filter, we must determine the values  $\tau_1$  through  $\tau_3$ . These are computed from  $T_1$  through  $T_3$  by

$$\begin{aligned}\tau_1 &= T_1 \\ \tau_2 &= T_2 - T_3 \\ \tau_3 &= T_3\end{aligned}\tag{9.75}$$

The elements  $(R, C)$  of the first filter section are now computed from  $\tau_1 = R_1 C_1$ ,  $\tau_2 = R_2 C_1$ , and  $\tau_3 = R_2 C_2$ .  $C_1$  can be chosen arbitrarily. It should be set such that the resistors are in a “reasonable” range—in other words, from 1 to 100 kΩ.

**Step 7 (Optional)** Sometimes the designer changes the values  $\tau_1$  through  $\tau_3$  and wants to know how the values  $T_1$  through  $T_3$  are altered. Then  $T_1$  through  $T_3$  can be computed from  $\tau_1$  through  $\tau_3$  from

$$\begin{aligned}T_1 &= \tau_1 \\ T_2 &= \tau_2 + \tau_3 \\ T_3 &= \tau_3\end{aligned}\tag{9.76}$$

The parameters of filter section 1 are determined now. Next, we must calculate the parameters of the second section. The procedure depends on the type of poles. When complex poles are desired, it continues with step 8, otherwise with step 10.

**Step 8 (Complex poles)** We must choose a suitable value for natural frequency  $\omega_s$  and damping factor  $\zeta_s$ . For  $\zeta_s$ , it is optimum to set  $\zeta_s = 0.707$ . Experience shows that sufficient phase margin is obtained if  $\omega_s$  is placed at  $5 \cdot \omega_3$ , hence we set  $\omega_s = 5\omega_3$ .

**Step 9** Given  $\omega_s$  and  $\zeta_s$ , we can determine the elements  $(R, C)$  of the second filter section. Because there are more unknown elements than equations,  $C_4$  can be chosen arbitrarily. It should be set such that we obtain “reasonable” values for the resistors—in other words in the range of 1 to 100 kΩ.

The remaining components are computed from

(9.77)

$$C_5 = \frac{2C_4}{\zeta_s^2}$$

$$R_4 = \sqrt{\frac{2}{\omega_s^2 C_4 C_5}}$$

$$R_5 = \frac{R_4}{2}$$

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

**Step 10** (Real poles) We must choose values for the break frequencies  $\omega_3$  and  $\omega_4$  now [cf. Fig. 9.24]. It is recommended to set  $\omega_4 = 5 \cdot \omega_3$  and  $\omega_5 = 5 \cdot \omega_4$ . We now can determine  $\tau_4 = 1/\omega_4$  and  $\tau_5 = 1/\omega_5$ .

**Step 11** To determine the elements of section 2, it is most practical to convert the filter function in Eq. (9.72a) into the filter function given in Eq. (9.72b)—thus, we would compute  $\omega_s$  and  $\zeta_s$  from the given  $\omega_4$  and  $\omega_5$ . Comparing coefficients in Eqs. (9.72a) and (9.72b) yields

$$\omega_s = \frac{1}{\sqrt{\tau_4 \tau_5}} \quad (9.78)$$

$$\zeta_s = \frac{\tau_4 + \tau_5}{2\sqrt{\tau_4 \tau_5}}$$

Given  $\omega_s$  and  $\zeta_s$ , we now can compute the  $R$  and  $C$  values using Eqs. (9.77).

## The Key Parameters of Higher-Order PLLs

In Chap. 3, we derived a number of equations for computing parameters such as lock-in range, pull-out range, and so on for second-order PLLs. The question arises now about how these parameters change with higher-order loops. As we have seen, the key parameters depended on quantities such as natural frequency, the damping factor, and gain factors such as  $K_d$ ,  $K_0$ , and so on.

When checking the magnitude plots for third- and higher-order PLLs, we note that some poles and zeroes are located at frequencies where the open-loop gain is greater than 1—for instance, that applies to the corner frequencies  $\omega_1$  and  $\omega_2$  in Fig. 9.3 (third-order loop). The additional poles are placed at frequencies where the open-loop gain is markedly less than 1 (cf. the corner at  $\omega_3$  in Fig. 9.3). These higher-frequency poles, therefore, have little impact on the transient behavior of the loop, hence they can be discarded for the calculation of lock-in and lock-out processes. Without these higher-frequency poles, the open-loop transfer function  $G(s)$  looks very much the same as the corresponding transfer function of a second-order loop. Consequently, we still can use the formulas listed in Tables 3.1 through 3.5 to compute key parameters such as lock-in range, lock-in time, and so on.

Although terms such as *natural frequency*  $\omega_n$  and *damping factor*  $\zeta$  are defined for second-order systems only, it has become customary to use such terms for higher-order PLLs as well. Doing so, we purposely discard the poles at those frequencies where the open-loop gain has dropped below 1. Because this is not fully correct mathematically, we should perhaps use the terms “*pseudo natural frequency*” and “*pseudo damping factor*” for those quantities.

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

# Computer-Aided Design and Simulation of Mixed-Signal PLLs

## Overview

A CD is distributed with this book that contains a program developed by the author, which enables the user to design and simulate PLLs of different kind. It is very easy to install the program. Just plug it into the CD player and installation will start automatically. If the install program does not start up, open Windows Explorer, click the D:\drive (it is assumed the CD uses drive D) and double-click the SETUP.EXE file in the right pane of Windows Explorer. During installation, follow the instructions on screen.

**Note** there has been a problem with users of German and French versions of Windows. When the *comma* symbol is used as a decimal point, the program crashes immediately after startup with an error message. To avoid this, the decimal point should be changed to a *period (dot)*.

The program can be started now. On startup, it displays a PLL block diagram that defines the symbols used throughout this application. In the toolbar (on top of the screen) are a number of speed buttons which look like this:



These buttons are used to call a number of options. Clicking the CONFIG button opens a configuration dialog ([Fig. 10.1](#)). This dialog box will be used to specify which category of PLL will be designed (Mixed Signal PLL or ADPLL). Moreover, the user will have to tell the program which types of phase detectors, loop filters, and so on will be used. Hitting the DESIGN speed button starts another dialog, the Filter Design dialog box ([Fig. 10.2](#)). Here, the user will have to enter key parameters such as 3 dB bandwidth ( $f_{3dB}$ ) and the like. This dialog

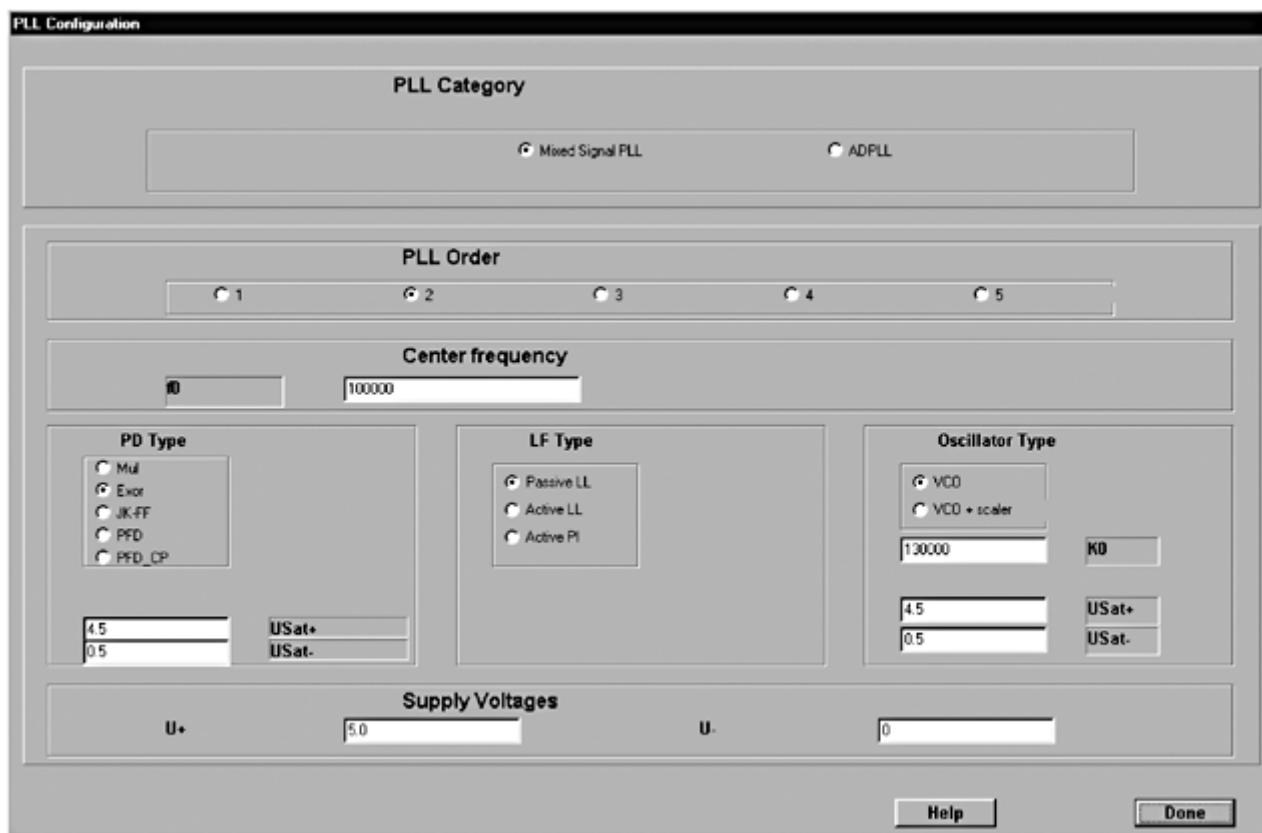
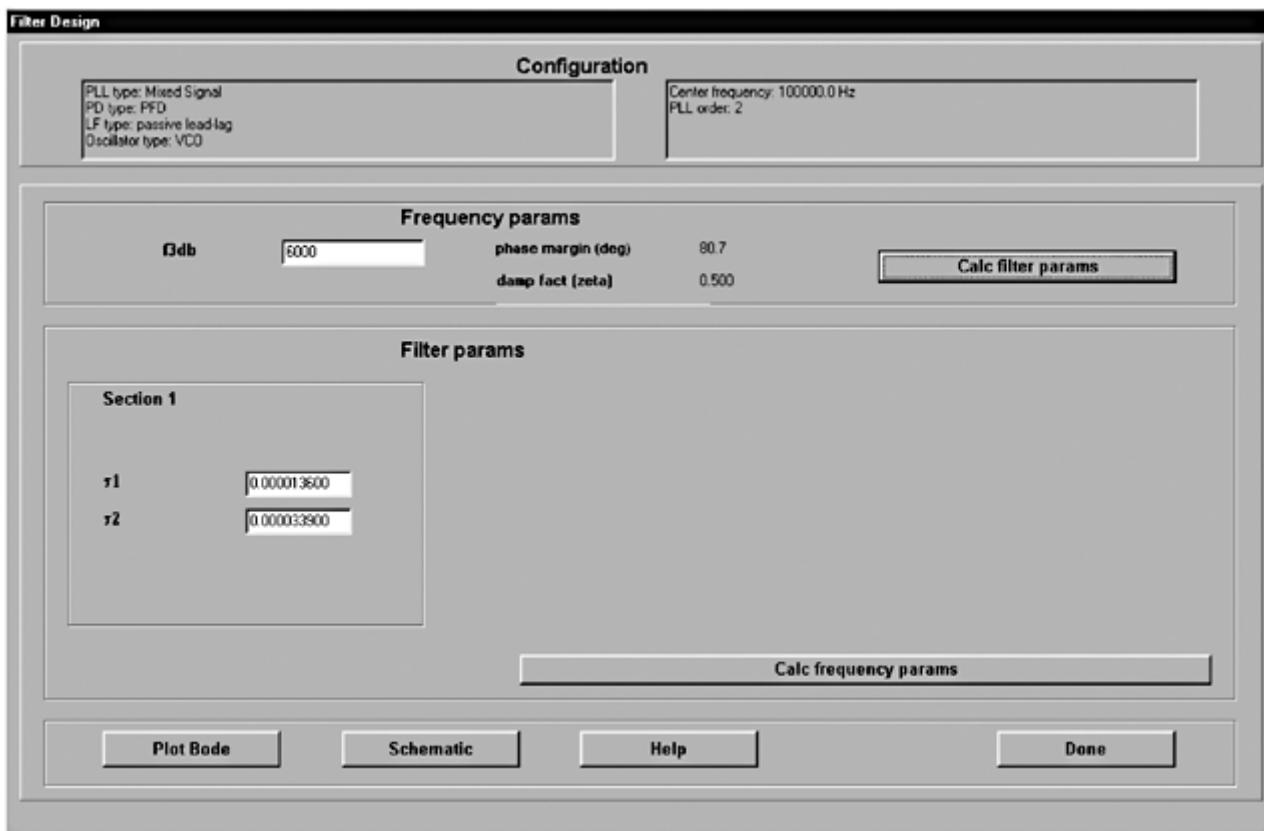


Figure 10.1 The dialog box for PLL configuration.



**Figure 10.2** The dialog box for the loop filter design.

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

box is used to compute the parameters of the loop filter (for example, time constants  $\tau_1$ ,  $\tau_2$ , and so on). With the SIM (simulation) speed button, the user enters simulation mode. Here, the response of the loop onto phase and frequency steps applied to the reference input can be simulated. The dialog box also allows the user to add noise to the reference signal. Using the OPT (options) speed button, the user starts an options dialog where he/she can shape the appearance of the graphical output frames and printouts individually. The user can, for example, choose colors for curves, background, grid, axis tick labels and so on, fonts, font size, font style (italic, bold) and make other choices. The speed button with the interrogation mark starts a help file, which provides plenty of information on this program, an overview on PLL fundamentals, and much more. The last speed button (EXIT) closes the program.

To see what the program can do, it is recommended you start the program and perform a quick tour.

## Quick Tour

Start the program. This brings up the block diagram of the PLL. At the screen's top is a toolbar with six speed buttons labeled CONFIG, DESIGN, and so on. Click the CONFIG button to start the system configuration.

### Configuring the PLL system

The dialog box for the CONFIG dialog is shown in [Fig. 10.1](#). On it are two radio buttons: Mixed Signal PLL and DPLL. Clicking the Mixed Signal PLL radio button simulates either an LPLL or a DPLL, while clicking the ADPLL radio button simulates an all-digital PLL (as described in [Chap. 11](#)). As you will see, the program starts with a default configuration when it runs the first time. When closing the program, all settings will be saved to a file (PARAMS.TXT). When the program is started again, it loads the last valid configuration from that file.

Below these two radio buttons are another five that specify the order of the PLL system being tested. The order number is identical with the number of poles of the system to be tested. PLL theory says that the order of the PLL is given by the order of the loop filter +1. Hence, a first-order PLL doesn't use a loop filter at all. A second-order PLL uses a first-order loop filter, and so on. For details, click the Help button in the CONFIG dialog box.

Below the five radio buttons is an edit control specifying the center frequency  $f_0$  of the PLL.

Having selected the center frequency, proceed with entering the parameters of the PLL's building blocks. If you checked the Mixed Signal PLL radio button, for example, you would now tell the program which type of phase detector to use. The panel labeled PD Type offers a choice of five different phase detectors: the multiplier PD, the EXOR PD, the JK-FF PD, the PFD with voltage output, and the PFD with charge pump output; the corresponding radio button is labeled PFD\_CP. Below the five radio buttons a number of edit controls show up;

depending on the type of PD chosen, there can be 1, 2, or 3 edit controls. These controls are used to enter the parameters of the selected phase detector. When a PFD with voltage output has been selected (as is the case in the example in Fig. 10.1), you must enter the saturation limits of the PFD (for example, 4.5 V for the upper, and 0.5 V for the lower saturation level).

In the panel LF Type, the desired type of loop filter must be entered. Again, when an active filter type has been specified, two edit boxes are displayed below the three radio buttons where the saturation limits of the loop filter must be entered.

In the panel Oscillator Type, the user must decide whether to use a VCO with or without a down scaler. Beneath the corresponding two radio buttons are a number of edit controls that hold the parameters of the VCO—for example, VCO gain  $K_0$  (in  $\text{rad s}^{-1}\text{V}^{-1}$ ).

To close the configuration dialog box, click the DONE button.

**Note** The CONFIG dialog box is a *modal* one—in other words, it can exit only by clicking the DONE button. This makes sure that only valid parameters can be entered into the edit controls: when the DONE button is clicked, a function is started that checks every entry for validity. First, numbers are tested for the correct format. If a user erroneously entered, say, “1.b” instead of “1.6” this is an illegal floating format and is therefore rejected. Second, the ranges of all entries are checked. For example, the positive saturation limit for the phase detector cannot be greater than the positive supply voltage, and so on.

## Designing the loop filter

Having closed the CONFIG dialog box, we now start the DESIGN dialog. Click the DESIGN speed button. This displays the DESIGN dialog box, as shown in Fig. 10.2. On top is a panel labeled Configuration, which shows the setup that has been performed in the CONFIG dialog. Below are two other panels. The upper is named Frequency Params, while the lower is labeled Filter Params.

When the DESIGN dialog box starts, the filter params (here  $\tau_1$  and  $\tau_2$ ) are taken from the values previously entered, or from the default values when the program runs the first time. Moreover, the program computes the frequency params (here  $f_{3\text{dB}}$ , phase margin, and damping factor) based on the given filter params  $\tau_1$  and  $\tau_2$ .  $f_{3\text{dB}}$  is the 3-dB corner frequency of the closed-loop gain of the PLL (in other words, the frequency where the gain is 3 dB below the DC gain). For details, see the Help file in the DESIGN dialog box.

You now can modify the filter in two ways:

- Enter an appropriate  $f_{3\text{dB}}$  value into the frequency params box, say, 5000 Hz, and then click the Calc Filter Params button. The program then calculates the values for  $\tau_1$  and  $\tau_2$  required to reach that goal. In addition, it also computes the values for the damping factor and phase margin.
- Alternatively, you can set the filter params ( $\tau_1$  and  $\tau_2$ ) directly. Having entered these values, hit the Calc Frequency Params button. The program then

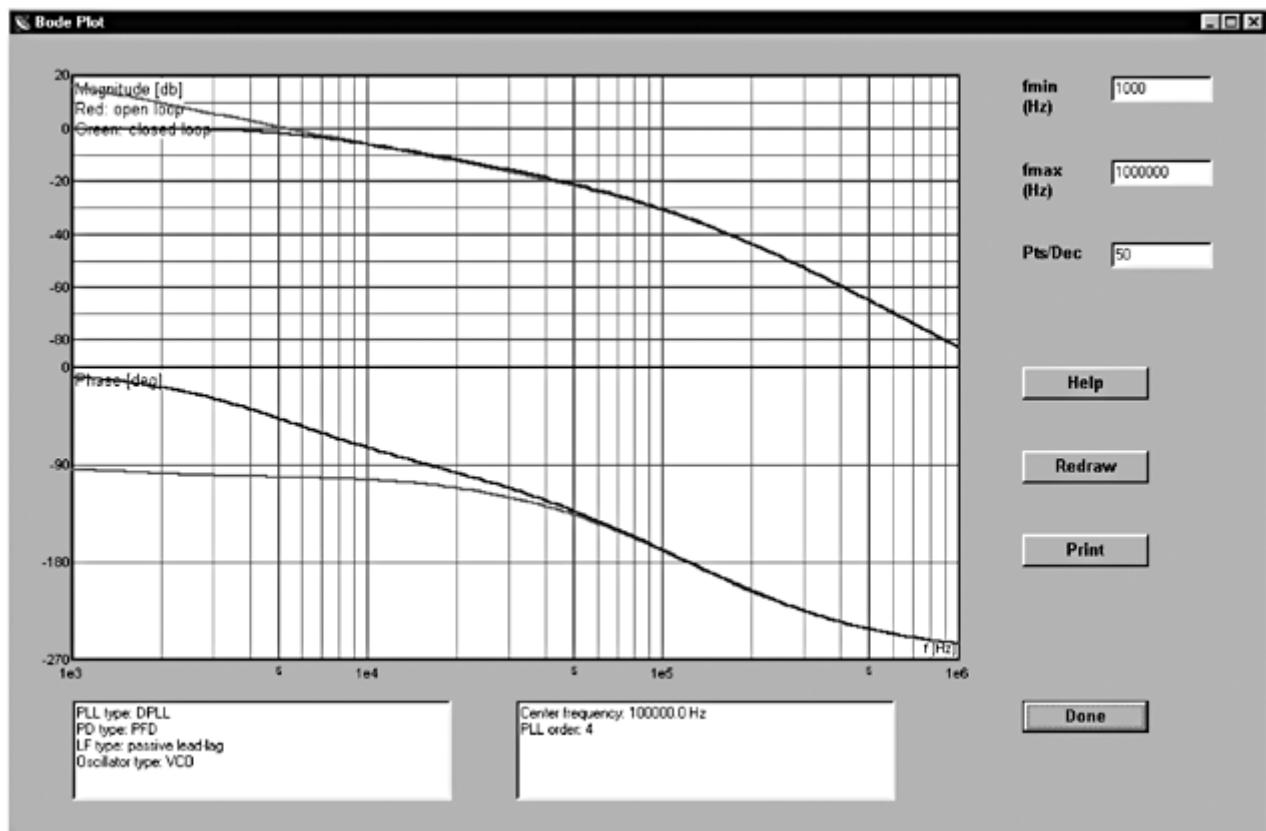
calculates the resulting values for  $f_{3\text{dB}}$ , phase margin, and damping factor. If you don't like that result, modify the filter params ( $\tau_1$  and  $\tau_2$  in this example) until the design is acceptable.

## Analyzing the stability of the loop

To analyze the stability of the loop, click the Plot Bode button, which is located at the bottom of the DESIGN dialog box (see Fig. 10.2). This brings up Bode plots for both open-loop and closed-loop phase transfer functions (see Fig. 10.3).

This is an example of a fourth-order loop. On screen, the curves are displayed in colors. The red curve is the open loop, while the green curve is the closed loop gain. In this text, however, the curves are printed in gray scale. To avoid confusing the open loop with the closed loop, remember that the closed-loop magnitude curve starts with about 0 dB. In the lower half, the phases are plotted. Because the system has four poles and one zero, the phase approaches  $-270^\circ$  at high frequencies. Here, the closed-loop phase curve starts with about  $0^\circ$ .

The transit frequency  $f_T$  is obtained from the crossover of the open-loop magnitude curve with the 0-dB line.  $f_T$  is approximately 5000 Hz. At that frequency, the phase (open-loop) is approximately  $-135^\circ$ , hence we have a phase margin of roughly  $45^\circ$ .



**Figure 10.3** The Bode diagram for open-loop and closed-loop gain of the PLL.

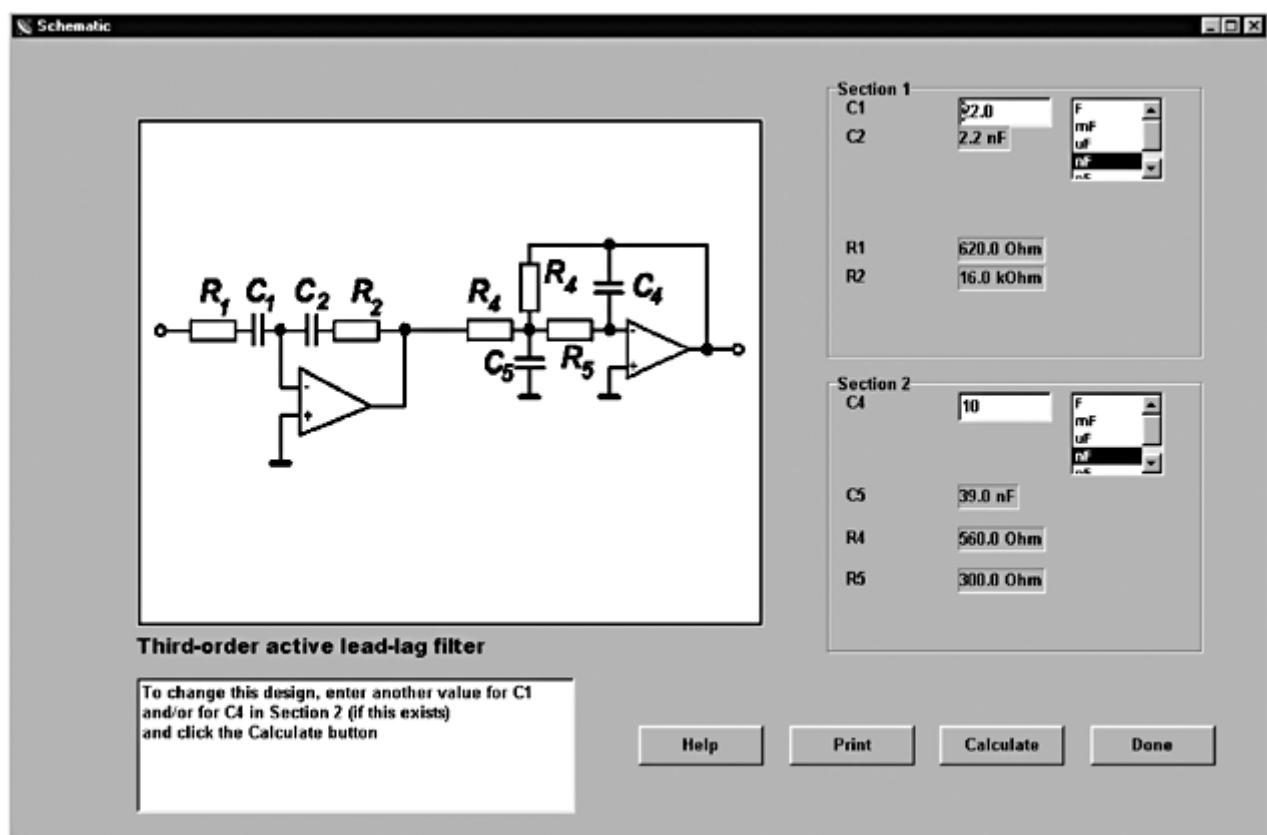
Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

Right from the curves, three edit windows are displayed: fmin, fmax, and Pts/Dec. fmin is the low end of the frequency scale, while fmax is the high end. The Bode plot is always formatted such that the low and high ends of the frequency scale are integer powers of 10. Thus, if you enter fmin = 4567 Hz, for instance, this is rounded downward to 1000, and if you specify fmax = 300 kHz, this is rounded upward to 1 MHz. Pts/Dec indicates how many points of the Bode curves are computed per decade of frequency. Default is 50 points per decade. Below the three edit controls are four pushbuttons in the Bode diagram dialog box: Help, Redraw, Print, and Done. If you entered different values for fmin, fmax, or Pts/Dec, click the Redraw button to recalculate and replot the new Bode diagram. You can print the Bode plots by hitting the Print button. Clicking the Done button closes the Bode diagram dialog box. The Help button brings up additional information about the Bode diagram.

## Getting the loop filter schematic

At the bottom of the DESIGN dialog box is a pushbutton labeled Schematic. Clicking this button opens the Schematic dialog box. The schematic of the loop filter is shown on the left side in [Fig. 10.4](#) and shows a third-order active lead-lag filter.

To the right of the schematic are some edit windows and some “static text” windows, which are not alterable. In this example, the program proposed



**Figure 10.4** A schematic of the loop filter.

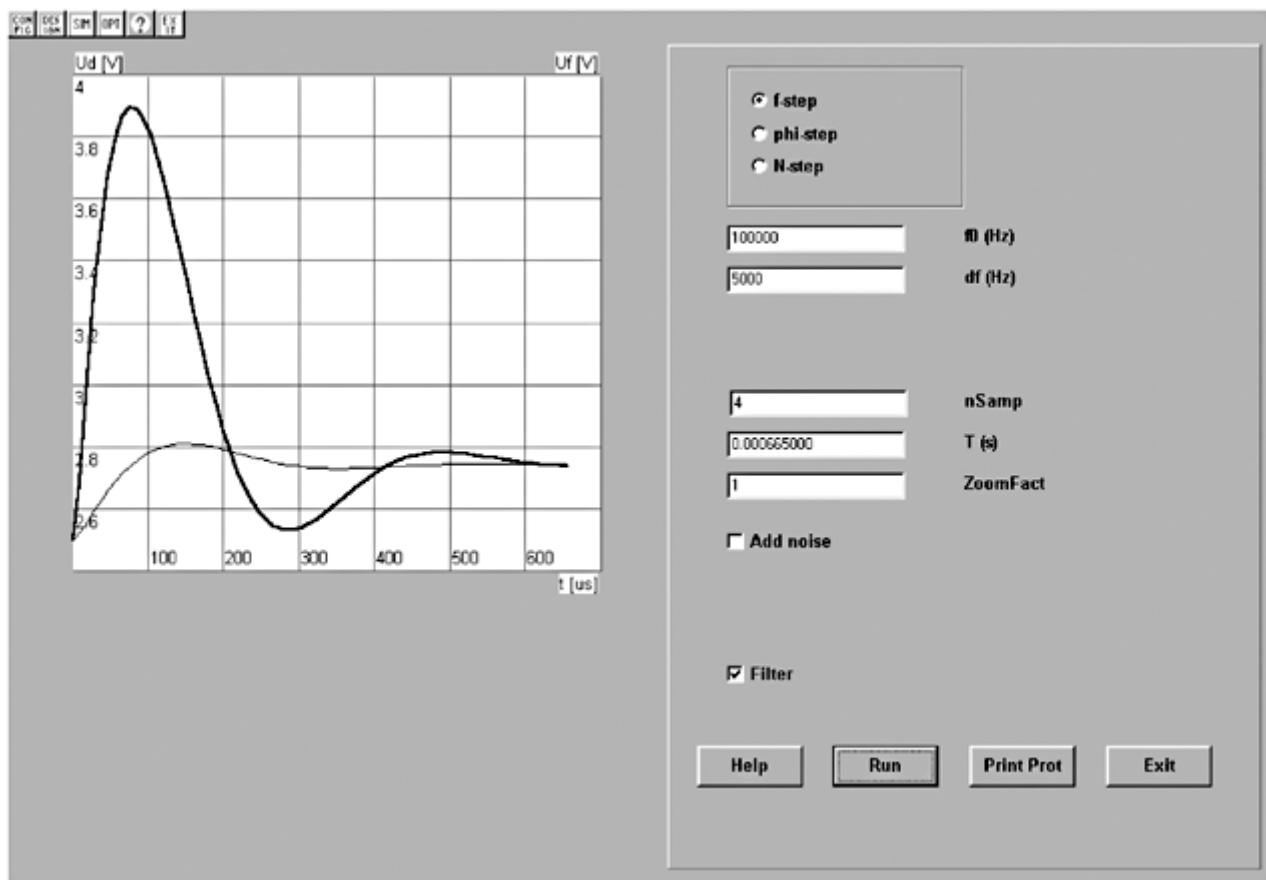
**Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.**

$C_1 = 22 \text{ nF}$  and  $C_4 = 10 \text{ nF}$ . The user can change these values, including the unit (F, mF,  $\mu\text{F}$ , nF, pF). Clicking the Calculate button restarts the computation of all the other elements ( $R$  and  $C$ ). These values are rounded to the closest value in the standard R24 series. (Note: the R24 series defines 24 standard values within one decade, so for resistors there are standard values of 1, 1.1, 1.2, 1.3, 1.5, 1.6, 1.8, 2.0, 2.2 ... up to 8.2, 9.1  $\Omega$ , and so on.) The values of  $C_1$  and  $C_4$  should be set such that the resulting resistor values have “reasonable values”—that is, in the range of 1 to 100 kohms. In the preceding example, the user may wish to get larger values for  $R_4$  and  $R_5$  and would probably enter a lower  $C_4$  (say, 1 nF). The schematic, including element values, can be printed out using the Print button. When the schematic is finished, click the Done button.

## Running simulations

We are still navigating through the DESIGN dialog box. Having completed our design, we can now proceed to simulations. To do so, close the design dialog by clicking the DONE button. Clicking the SIM speed button enters the simulation dialog box, as shown in Fig. 10.5.

In Fig. 10.5, a number of simulation parameters can be entered. On top of the dialog box are three radio buttons: f-step, phi-step, and N-step. Checking



**Figure 10.5** Simulation of a frequency step applied to the reference input.

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

the f-step simulates a frequency step applied to the reference input of the PLL. When doing so, the size of the frequency step must be entered into the df edit window (5000 Hz in this example). When the phi-step radio button is checked, the df edit window is blanked, and another edit window labeled dphi appears. In this case, a phase step applied to the reference input is simulated. Lastly, when the N-step radio button is checked, the program simulates a change of the divider ratio of the down scaler. With this option, the user can simulate an integer-N frequency synthesizer where the division ratio of the down scaler is abruptly changed in order to get another output frequency. The edit window nSamp specifies the number of samples to be calculated in one reference cycle of the PLL simulation; the default value is 4. Note that there is a checkbox labeled Filter in this dialog box. This control is checked by default. With nSamp = 4 and with the checkbox enabled, the program computes the average of the four samples in every reference cycle, which means that the curves plotted on the left side do not look like the waveforms you would get from an oscilloscope but rather an averaged plot. Such “oscilloscope” plots are available, however, by specifying larger values for nSamp, which will be discussed in [Sec. 10.3](#).  $T$  is the duration of the simulation, which is 665  $\mu$ s in the example. Lastly, a ZoomFactor can be specified to zoom in on the results of the simulation. When ZoomFact = 10, for instance, the time axis is zoomed in by a factor of 10. You can browse through the results by moving the thumb of a scrollbar. When the Add Noise box is checked, noise is added to the input signal of the PLL being tested. This will be demonstrated in [Sec. 10.4](#).

To start the simulation, click the RUN button. The result of that simulation is shown in the left half of the dialog box. This is the response of the PLL onto a frequency step  $df = 5$  kHz applied to the reference input. The signals  $u_d$  and  $u_f$  are displayed versus time. On the screen, the curves are in color (in this book they are printed in gray scale). To see which curve represents  $u_d$  and which signifies  $u_f$  the  $u_d$  signal is plotted with a larger line width. This convention will be used in the following simulations.

To see what happens when specifying different types of phase detectors, loop filters, and so on, go back to the CONFIG and/or DESIGN dialog boxes, make the necessary modifications, and return to SIMUL again.

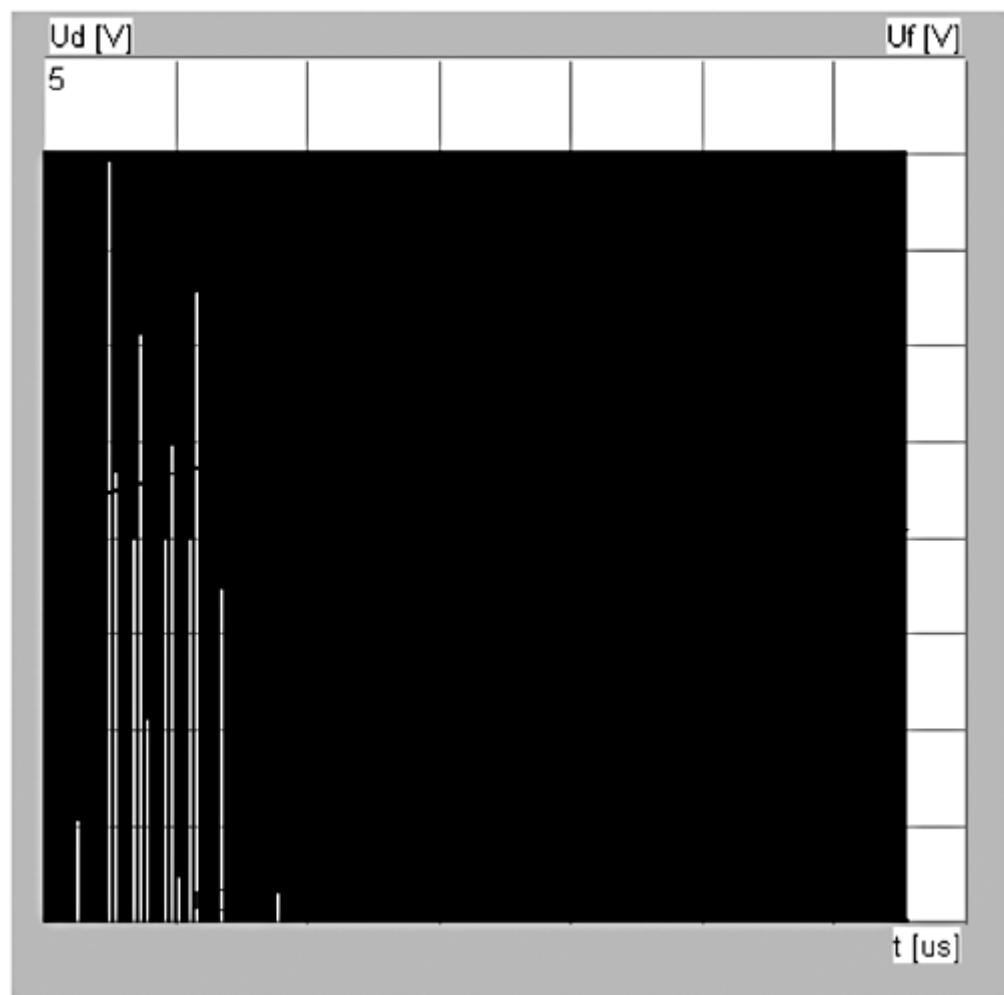
## Simulations with and without Averaging

When the parameter nSamp was chosen as 4 with the Filter checkbox enabled, you probably noticed that the curves obtained from the simulations look “nicer” than those you would get when measuring transients of a real PLL system with an oscilloscope. Because the designer of a PLL is mainly interested in the average  $u_d$  and  $u_f$  signals, the higher-frequency components are discarded in most cases. The simulation program also discards those AC components: under normal operation, it filters out the higher-frequency terms. By default, the simulation program computes four samples for  $u_d$  and  $u_f$  in each reference

cycle (when the PFD with current output is chosen,  $i_d$  is computed instead of  $u_d$ ). The value 4 is dictated by Nyquist's sampling theorem. Assuming that the frequency of the reference signal is  $f_1$  and the multiplier type phase detector is used, the output signal of that phase detector contains an AC component at  $2 \cdot f_1$ . To satisfy the sampling theorem, we must sample that signal using  $4 \cdot f_1$  at least.

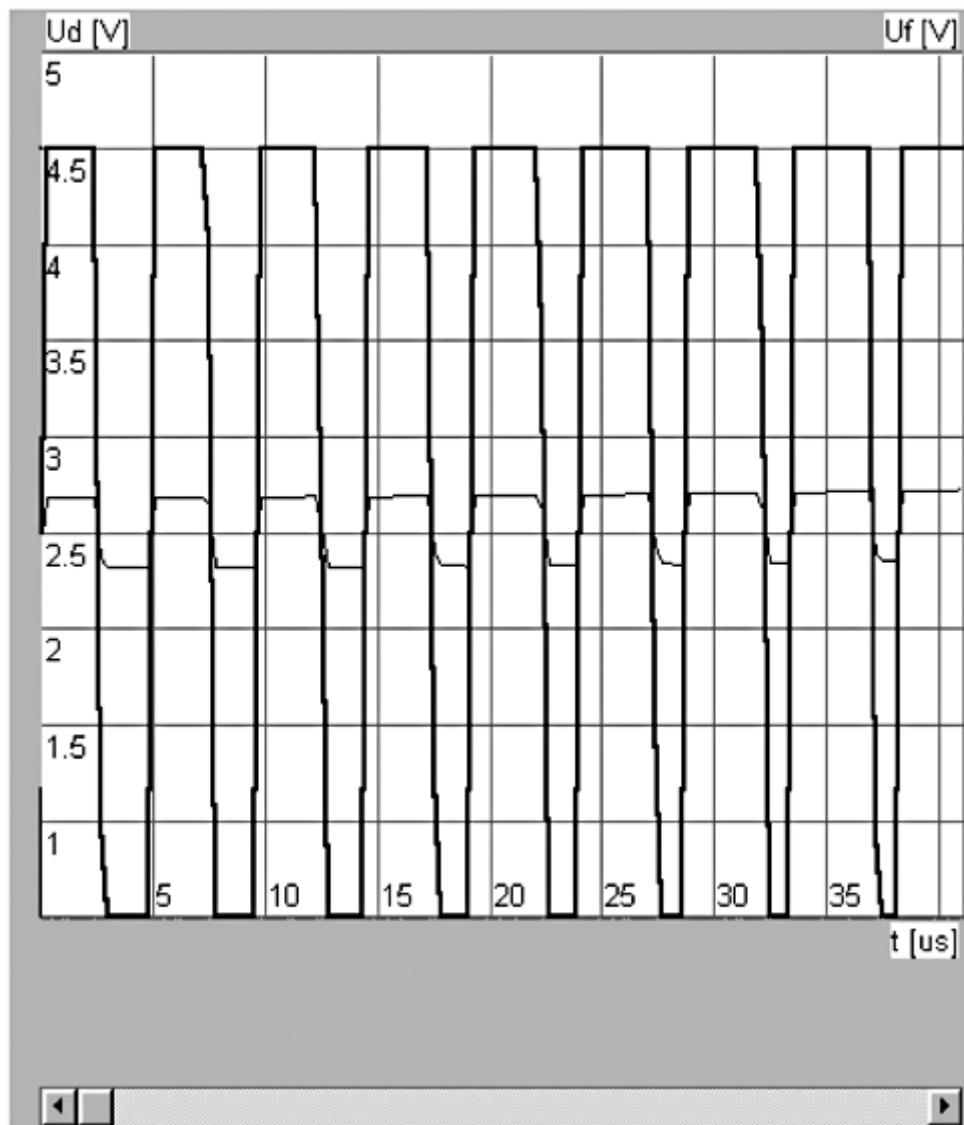
There are cases, however, where the user wants to see how the waveforms look in reality. This is done by unchecking the Filter checkbox. When only four samples are computed in each reference period, however, this gives a very crude approximation of the real waveform. To get better signals, we would increase the number of samples nSamp. nSamp can be set, therefore, in the range of 4 to 64. We repeat the simulation done in [Sec. 10.2.5](#) with the Filter checkbox unchecked and nSamp = 32 (see [Fig. 10.6](#)).

That looks pretty bad indeed! The curves are "smeared" over the full window. To see what really happens, we will zoom in on the display. With a zoom factor of 16, the time scale is expanded by a factor of 16. Below the frame window is



**Figure 10.6** Response on  $df = 5$  kHz, no filtering, nSamp = 32.

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.



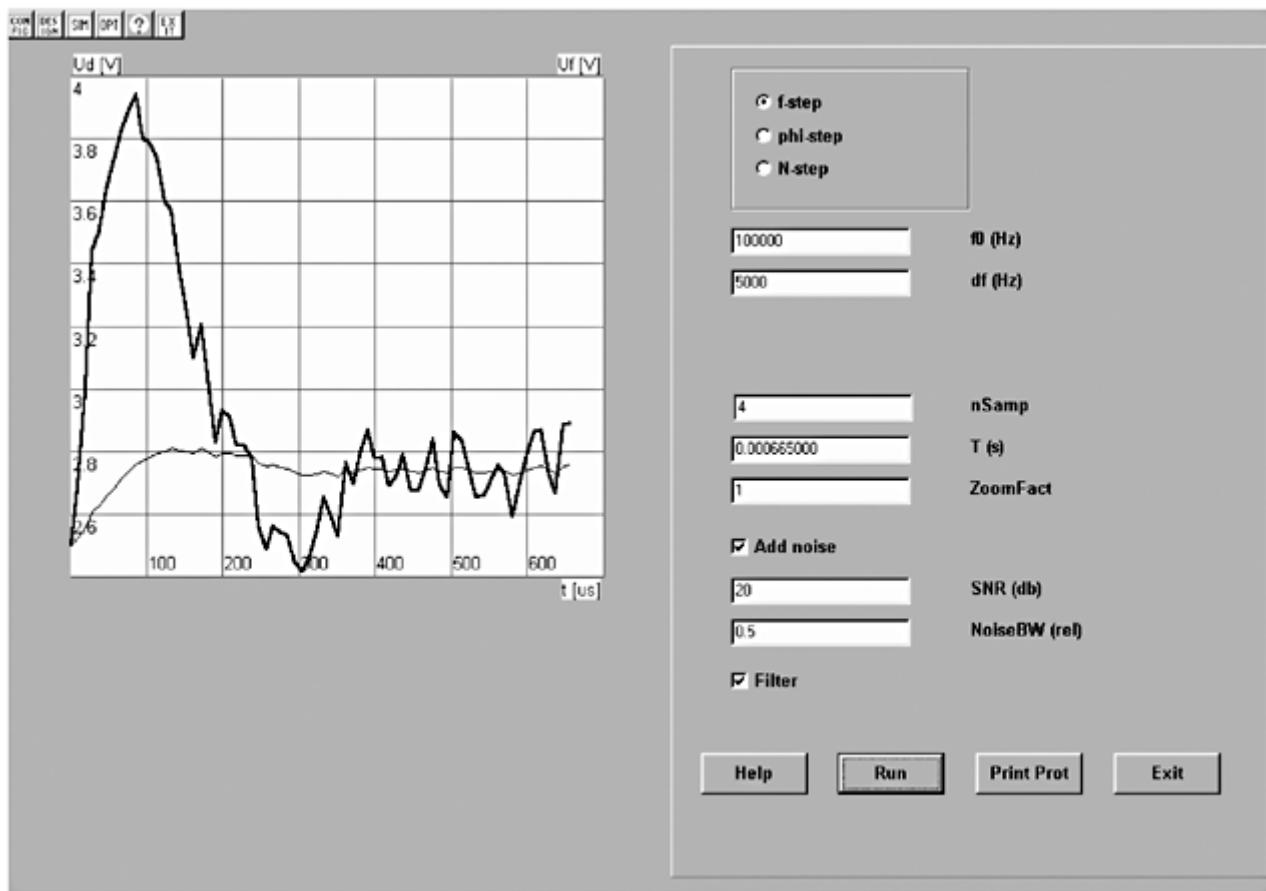
**Figure 10.7** Response on  $df = 14 \text{ kHz}$ , no filtering, nSamp = 32, zoom factor = 16.

a scrollbar that allows the user to scroll through the time window. Now we clearly see what the waveforms really are (see Fig. 10.7).

## Simulations with Noisy Reference Signals

The simulation program also allows you to add noise to the reference signal. To do so, check the AddNoise checkbox, as shown in Fig. 10.8. This opens two additional edit controls: SNR (dB) and NoiseBW(rel). SNR(dB) is the signal-to-noise ratio at the reference input, and NoiseBW(rel) is the (one-sided) noise bandwidth related to the (scaled-down) center frequency  $f_0'$ . In the example shown, SNR is chosen as 20 dB, while NoiseBW(rel) is set to 0.5. Because the scaled-down center frequency is 100 kHz, the one-sided noise bandwidth is

50 kHz now, which means there is a broadband noise whose power density spectrum reaches



**Figure 10.8** Response to  $df = 5$  kHz, with superimposed noise.

from 50 to 150 kHz. The Help file contains a lot of additional information on noise.

Using the parameters entered in this window, the result of the simulation is shown in Fig. 10.8. The curves have become “jittery” now, but it is clearly seen that the PLL perfectly locks nevertheless. The noise on the  $u_d$  waveform is much larger than the noise on the filtered signal  $u_f$  because the filter bandwidth is much less than the bandwidth of the noise spectrum.

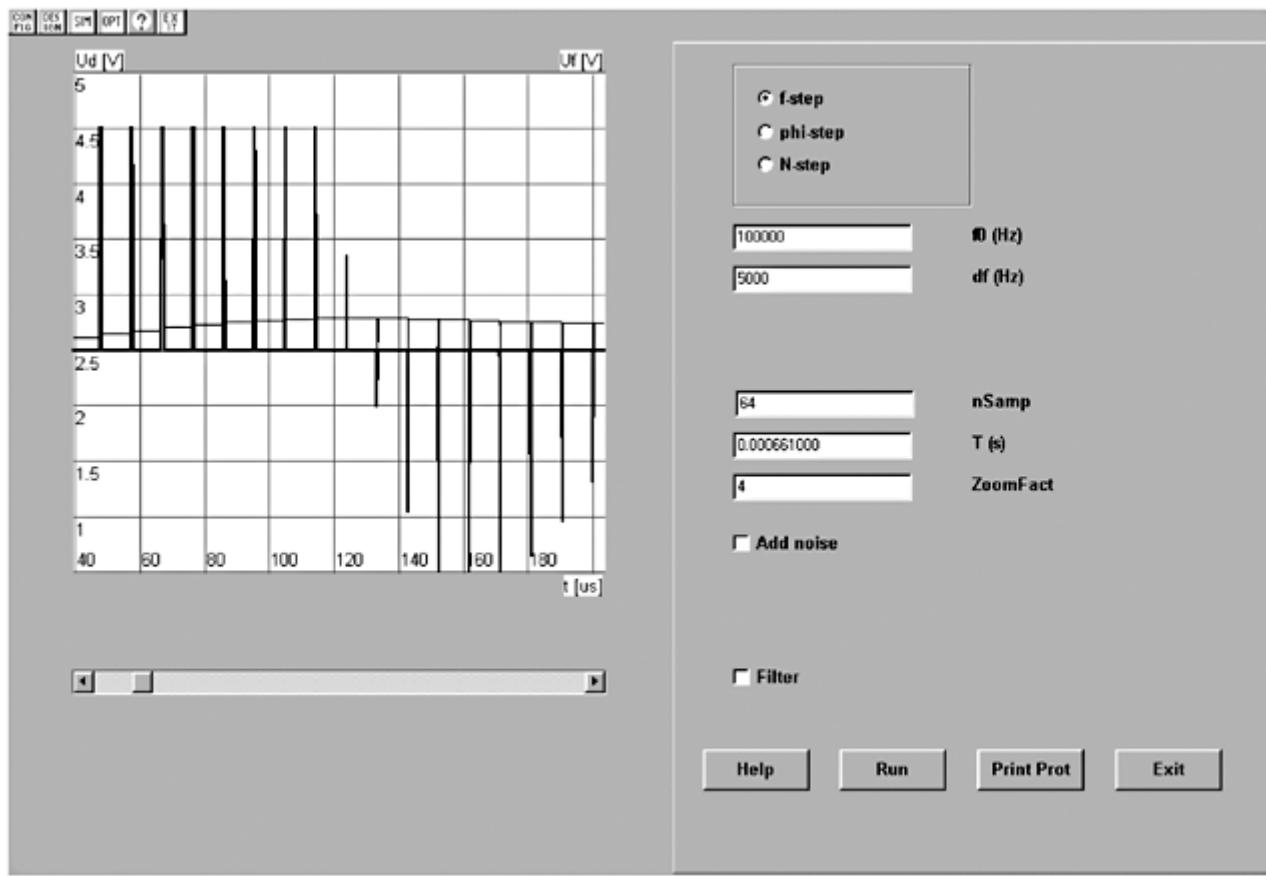
## Displaying Waveforms of Tri-State Signals

As we know, the output signal of the PFD with voltage output is a tri-state signal. It can be either HIGH or LOW, or it can be in the high-impedance state (high-Z). Logical HIGH and LOW levels can be easily displayed, but what about the high-Z state? In the simulation program, the high-Z state is represented by a signal level which is the mean of positive and negative supply voltages. When a circuit is powered from a unipolar 5 V supply, for example, the high-Z state is represented by 2.5 V. Figure 10.9 shows an example.

The time axis is zoomed in by a factor of 16 here. The signal  $u_d$  is in the high-Z state most of the time. At the left of the time scale, it temporarily switches to the HIGH state, while at the right it switches to the LOW state.

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.



**Figure 10.9** Simulation of an acquisition process using the PFD with tri-state (voltage) output.

## Getting Help

The help text in this program is context-sensitive. When the speed button with the interrogation mark in the toolbar is clicked, the content page of the help file is displayed. Help can be called from a number of different locations. If, for example, the Help button in the Bode window is clicked ([Fig. 10.3](#)), the corresponding Help item is immediately shown.

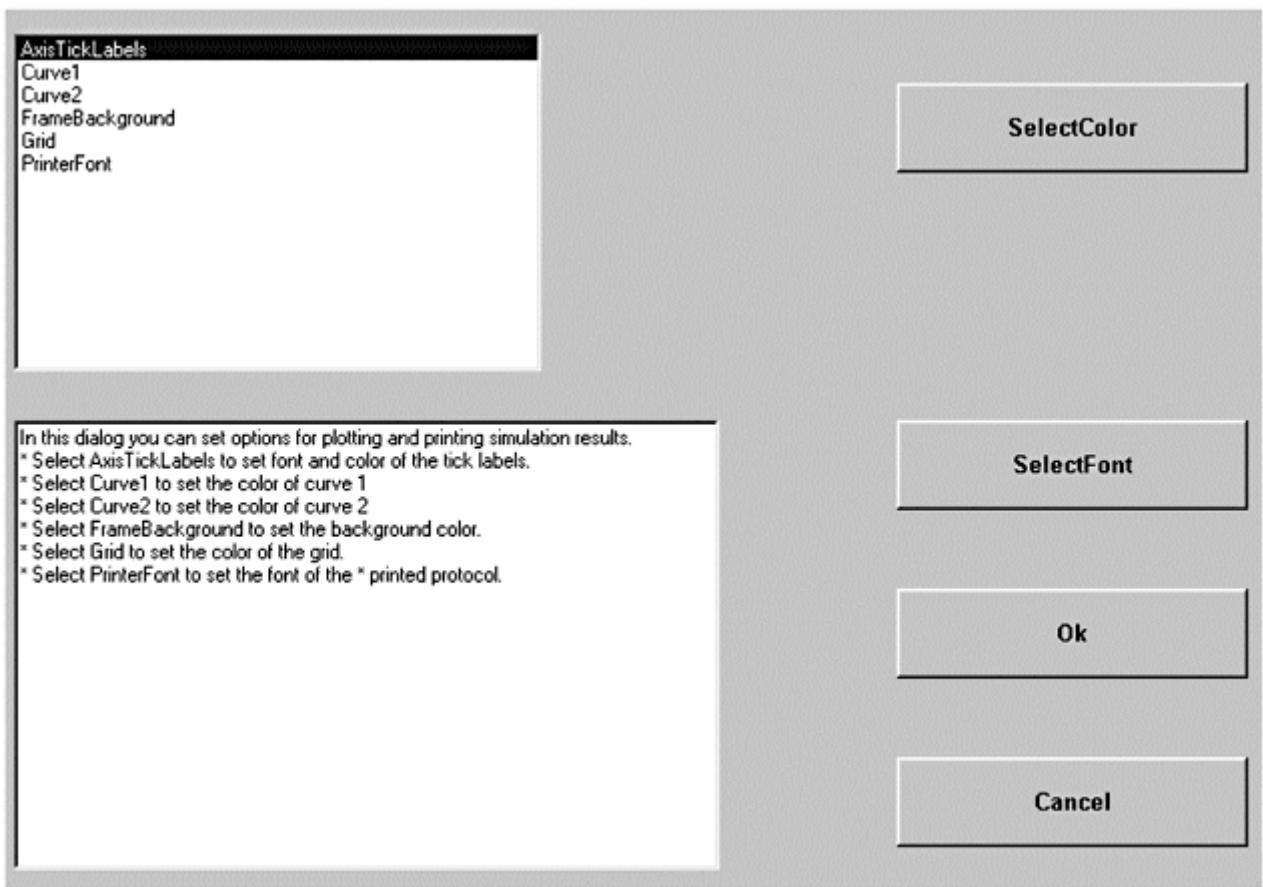
## Shaping the Appearance of Graphic Objects

The user can select a number of options for displaying and printing results of simulations. When the OPT speed button is clicked, an options window is displayed (see [Fig. 10.10](#)).

At the top left of the window is a list box containing six items. In the example, AxisTickLabels is highlighted. For these tick labels, the user can select the color and the font (with all its attributes, such as font size, font style, and so on). The Curve1 option refers to the display of  $u_d$  versus time. Here, the operator can select line color, line width, and so on. All selected parameters are stored in the configuration file when the program exits. They remain valid when the program is started next.

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.



**Figure 10.10** The Options window.

## Suggestions for Case Studies

Familiar with the simulation program, now try to simulate other kinds of PLLs. The following are some suggestions:

**Case study 1:** *Influence of loop filter type.* Use the active lead-lag instead of the passive, specify a DC gain  $K_a$  greater than 1, and see how the key parameters such as pull-out range and pull-in range are influenced. Use the active PI filter then. What is the major difference compared with other filter types? What happens to the pull-in range?

**Case study 2:** *Phase step.* Apply a phase step to the reference input. Vary the size of the phase step.

**Case study 3:** *A VCO with a divide-by-N counter.* Simulate a PLL whose output frequency is  $N$  times larger than the reference frequency. Check how the damping factor  $\zeta$  depends on the scaling factor  $N$ .

**Case study 4:** *Pull-in processes.* Using different types of loop filters, measure the pull-in time of the PLL under various conditions (different values for  $\zeta$ ,  $K_0$ ,  $K_d$ ,  $K_a$ , and the different

size of frequency step  $\Delta f_1$ ). Compare the results of the simulation with the values predicted by theory; use the formulas in Tables 3.1 through 3.5 to compute pull-in time  $T_P$ . Where can you find the largest deviations between theory and practice?

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

# All-Digital PLLs (ADPLLs)

## ADPLL Components

As we have seen in [Chap. 2](#), the classical “digital PLL” (DPLL) is a semi-analog circuit. Because it always needs a couple of external components, its key parameters will vary because of parts spread. Even worse, the center frequency of a DPLL is influenced by parasitic capacitors on the DPLL chip. Its variations can be so large that trimming can become necessary in critical applications. Many parameters are also subject to temperature drift and aging.

The all-digital PLL (ADPLL) does away with these analog-circuitry headaches. In contrast to the older DPLL, it is an entirely *digital* system. Let’s note first that the term “digital” is used here for a number of different things. First of all, “digital” means that the system consists exclusively of logical devices. But “digital” also signifies that the signals within the system are digital, too. Hence a signal within an ADPLL can be a binary signal (or “bit” signal), as was the case with the classical DPLL, but it can just as well be a “word” signal—that is, a digital code word coming from a data register, from the parallel outputs of a counter and the like. When discussing the various types of ADPLL, we find the whole palette of such digital signals.

To realize an ADPLL, all function blocks of the system must be implemented by purely digital circuits. Digital versions of the phase detector are already known, but we now have to find digital circuits for the loop filter and for the voltage-controlled oscillator (VCO), too. As we will see in [Sec. 11.1.3](#), the digital counter-part of the VCO is the digital-controlled oscillator (DCO). An almost unlimited number of purely digital function blocks are available for the ADPLL. To save space, we shall concentrate on those most frequently used.

## All-digital phase detectors

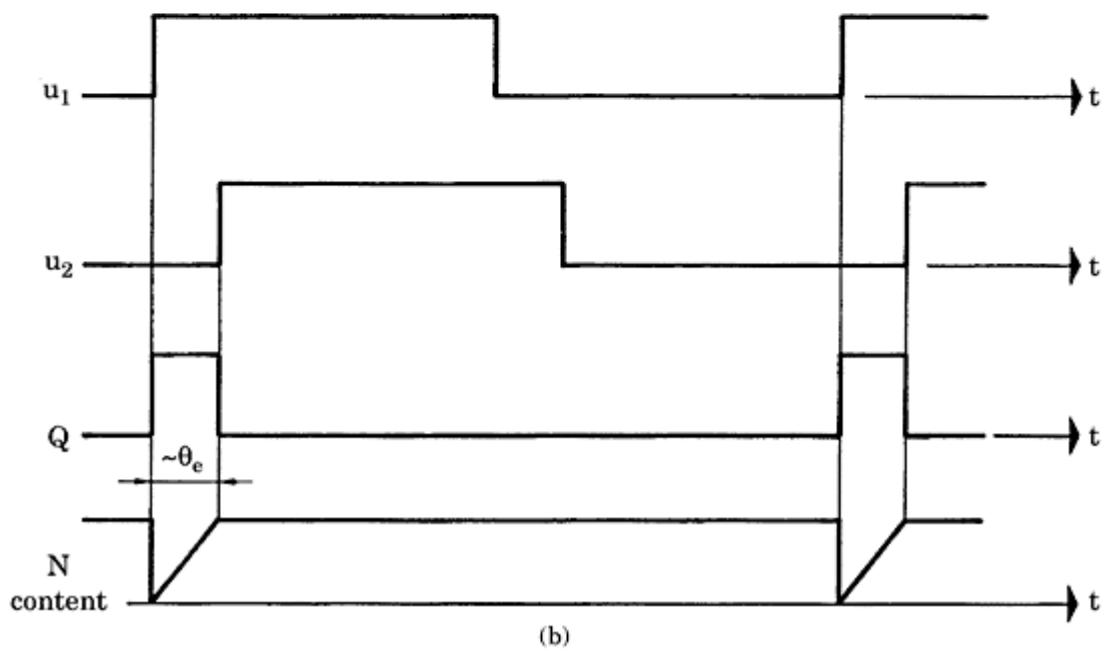
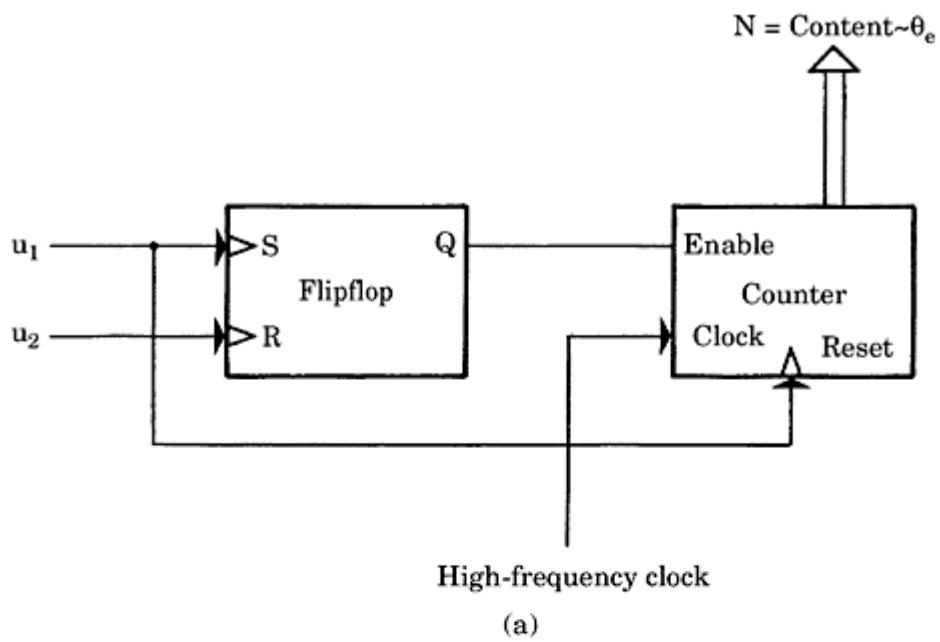
The three most important digital phase detectors have already been discussed in [Sec. 2.4](#). When digital word signals instead of bit signals are used, a number of additional phase-detector circuits become available.

---

A logical evolution of the simple JK-flipflop PD is the FF-counter phase detector illustrated in [Fig. 11.1a](#). The corresponding waveforms are shown in [Fig. 11.1b](#).

The reference (input) signal  $u_1$  and the output (or scaled-down output) signal  $u_2$  of the DCO (or VCO) are binary-valued signals. They are used to set or reset an edge-triggered RS-flipflop. The time period in which the  $Q$  output of the flipflop is a logic 1 is proportional to the phase error  $\theta_e$ . The  $Q$  signal is used to gate the high-frequency clock signal into the (upward) counter. Note that the counter is reset on every positive edge of the  $u_1$  signal.

The content  $N$  of the counter is also proportional to the phase error  $\theta_e$ , where  $N$  is the  $n$ -bit output of this type of phase detector. The frequency of the high-frequency

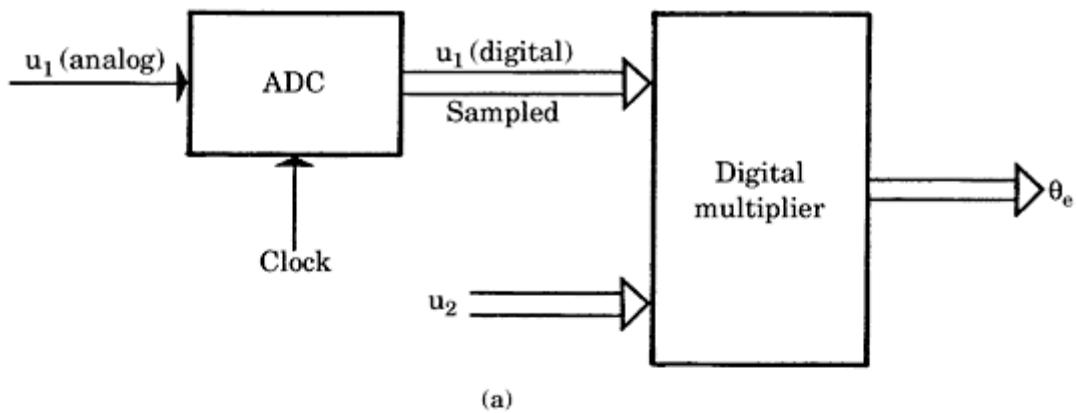


**Figure 11.1** A flipflop-counter PD. (a) A block diagram. (b) Corresponding waveforms.

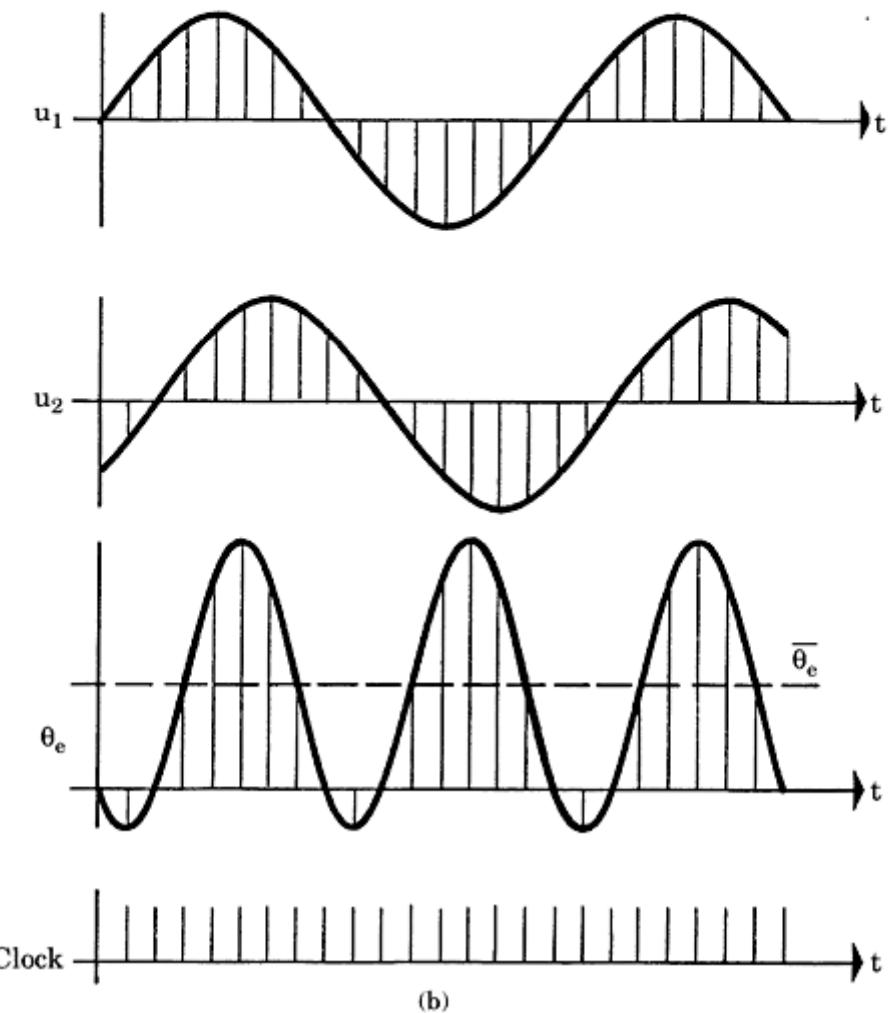
---

clock is usually  $Mf_0$ , where  $f_0$  is the frequency of the reference signal and  $M$  is a large positive integer.

Another all-digital phase-detector circuit has become known by the name Nyquist rate phase detector (NRPD).<sup>8</sup> The name stems from the well-known Nyquist theorem which states that a continuous signal can be reconstructed from a sampled version only if the sampling rate is at least twice the highest-frequency component of the signal. The block diagram of the NRPD is shown in Fig. 11.2a, and the corresponding waveforms are seen in Fig. 11.2b.



(a)



(b)

**Figure 11.2** The Nyquist rate PD. (a) A block diagram. (b) Corresponding waveforms.

The input signal is supposed to be an analog signal. It is periodically sampled and digitized at the clock rate. In the example shown, the clock rate chosen is 16 times the signal frequency, which is eight times more than required by the sampling theorem. The signal  $u_2$  is an  $N$ -bit digital word, generated by a DCO. (Refer also to Sec. 11.1.3.) Furthermore, the signal  $u_2$  has been drawn as a sine wave in Fig. 11.2b; another waveform (such as a square wave) could be used as well. The digitized signal  $u_1$  and the signal  $u_2$  are multiplied together by a digital multiplier. Thus, the NRPD operates similarly to the multiplier PD (type 1), as discussed in Sec. 2.4.1. This multiplier can be a hardware device, or the multiplication can be done by software, for example, in a microcontroller. The resulting phase error signal  $\theta_e$  is also shown in Fig. 11.2b. Its average value must be filtered out by a succeeding digital loop filter, as will be demonstrated in Sec. 11.1.2.

Still another method of measuring the phase error is the *zero-crossing technique*.<sup>8</sup> The simplest zero-crossing phase detector is illustrated in Fig. 11.3a (its waveforms are shown in Fig. 11.3b). The reference signal  $u_1$  is supposed to be analog;  $u_2$  is a binary signal. The positive transitions of  $u_2$  are used to clock the analog-to-digital converter (ADC), so  $u_1$  is sampled once during every reference period. The digital output signal of the ADC is then proportional to the phase error. Usually, this signal is held in a buffer register until the next conversion is completed; thus, the phase error signal  $u_e$  is a quasi-continuous signal, as shown by the dashed line in Fig. 11.3b.

Figure 11.4 shows another variant of digital PD, the so-called *Hilbert-transform PD*. The key element of this PD is a Hilbert transformer. This is a special digital filter that shifts the phase of a sinusoidal input signal by exactly  $-90^\circ$  irrespective of its frequency.<sup>13</sup> Moreover, the gain of the Hilbert transformer is 1 at all frequencies. In the block diagram of Fig. 11.4a, the Hilbert transformer is shown in the top left corner and is marked by the symbol  $-\pi/2$ . Assuming the input of the device is a digital word signal of the form

$$u_1(t) = \cos(\omega_0 t + \theta_e) \quad (11.1)$$

the output signal of the Hilbert transformer is given by

$$\hat{u}_1(t) = \cos\left(\omega_0 t + \theta_e - \frac{\pi}{2}\right) = \sin(\omega_0 t + \theta_e) \quad (11.2)$$

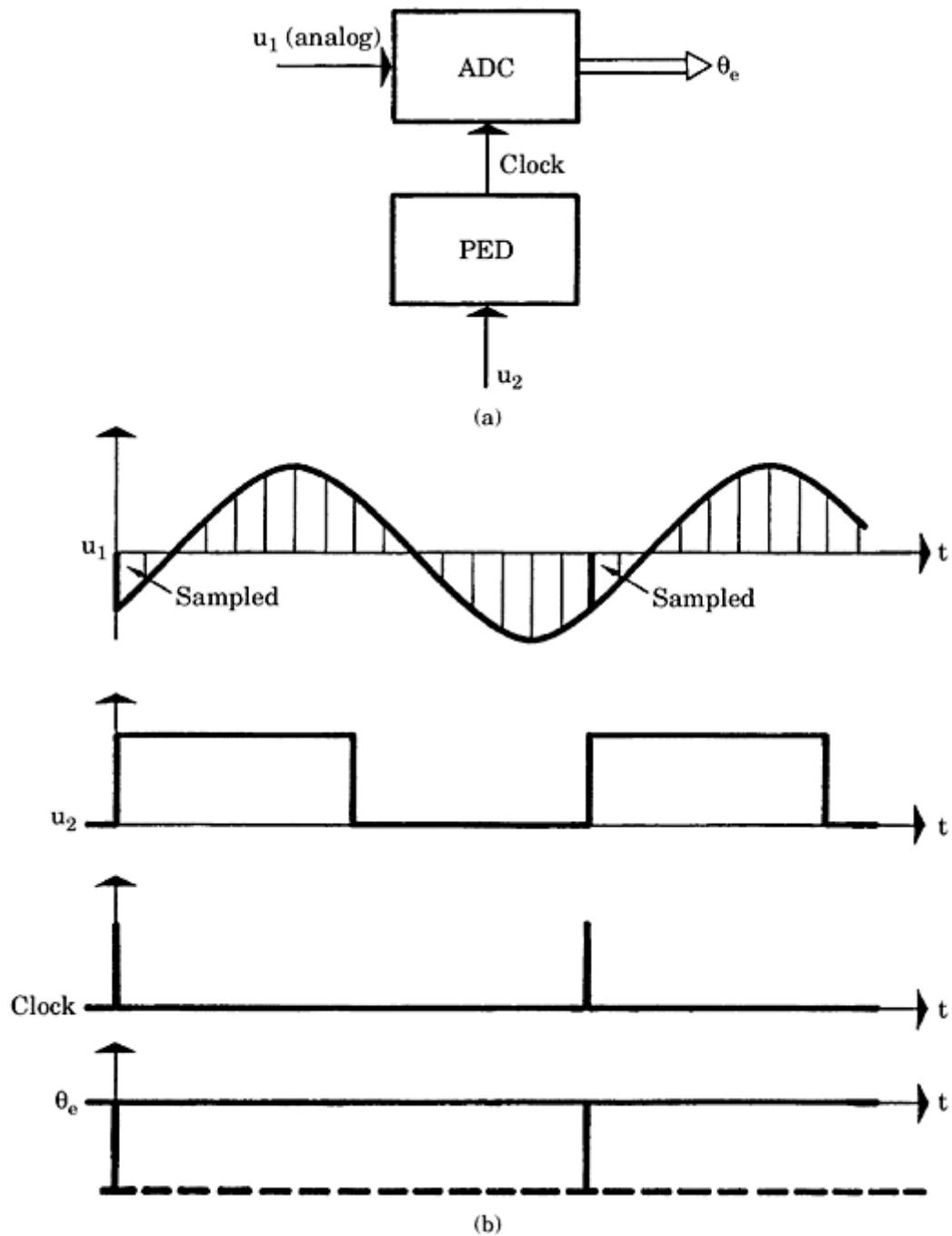
The Hilbert-transform PD extracts the phase error  $\theta_e$  by trigonometric computations, which are shown in the following. The DCO used in this type of phase detector is supposed to generate two output signals, an *in-phase* signal  $I = \cos(\omega_0 t)$  and a *quadrature* signal  $Q = \sin(\omega_0 t)$ .

By the trigonometric operations

$$\begin{aligned} \cos \theta_e &= I u_1 + Q \hat{u}_1 \\ \sin \theta_e &= I \hat{u}_1 - Q u_1 \end{aligned} \quad (11.3)$$

the sine and cosine of the phase error  $\theta_e$  are computed. Dividing the sine by the cosine term

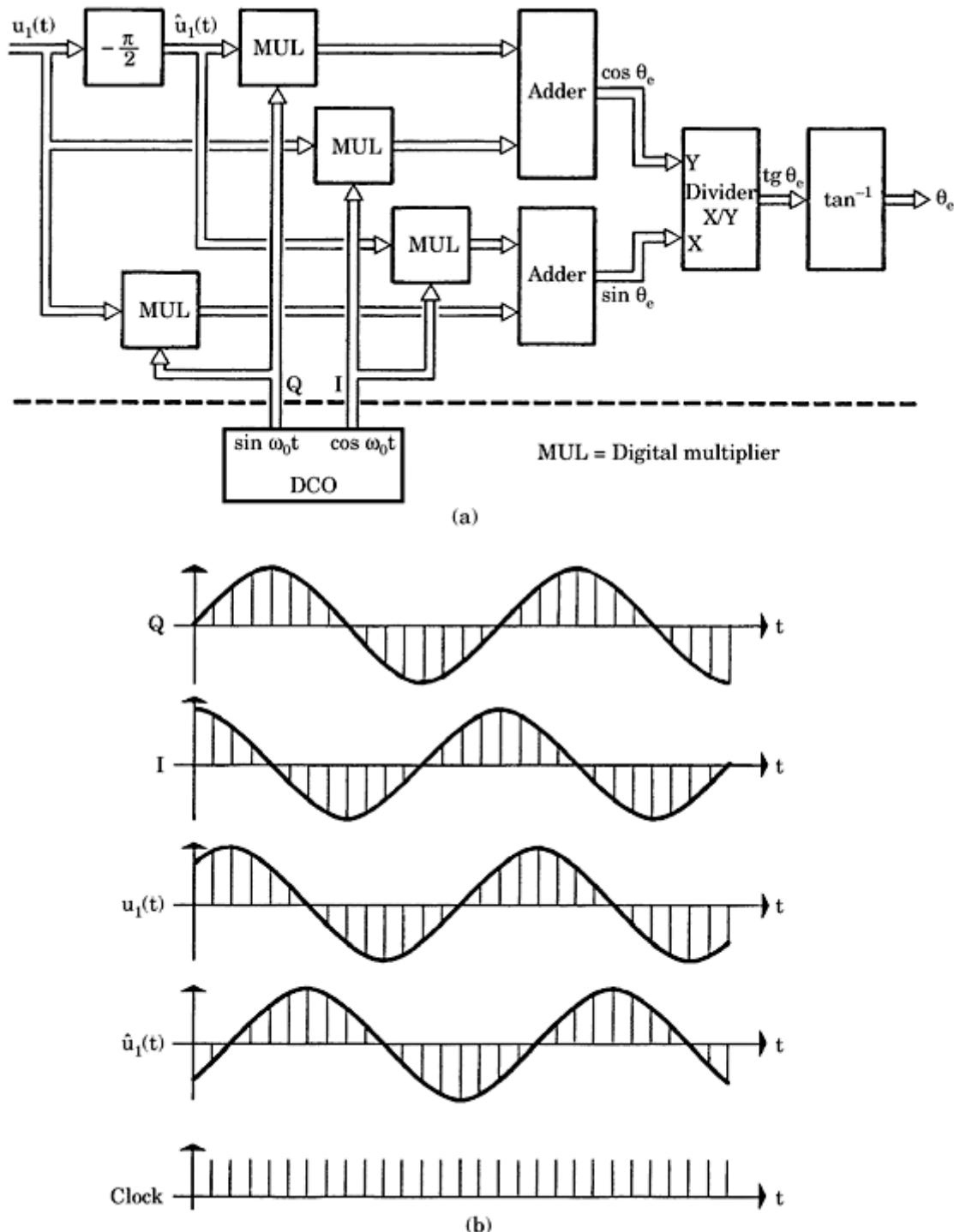
yields the tangent of phase error, while using a digital algorithm



**Figure 11.3** A zero-crossing PD sampling the phase error at the positive transitions of the reference signal. (a) A block diagram (PED = positive-edge detector). (b) Corresponding waveforms.

for the inverse tangent ( $\tan^{-1}$ ), the phase error is obtained. The double lines in Fig. 11.4a signify that all signals within this device are digital word signals. The signals of the Hilbert-transform PD are shown in Fig. 11.4b. As in the NRPD, all mathematical computations are performed under the control of a clock signal whose frequency is usually  $M$  times the signal

frequency  $f_0 = \omega_0/2\pi$ . The variety of mathematical operations required by the Hilbert transform strongly suggests implementation of this method by software.

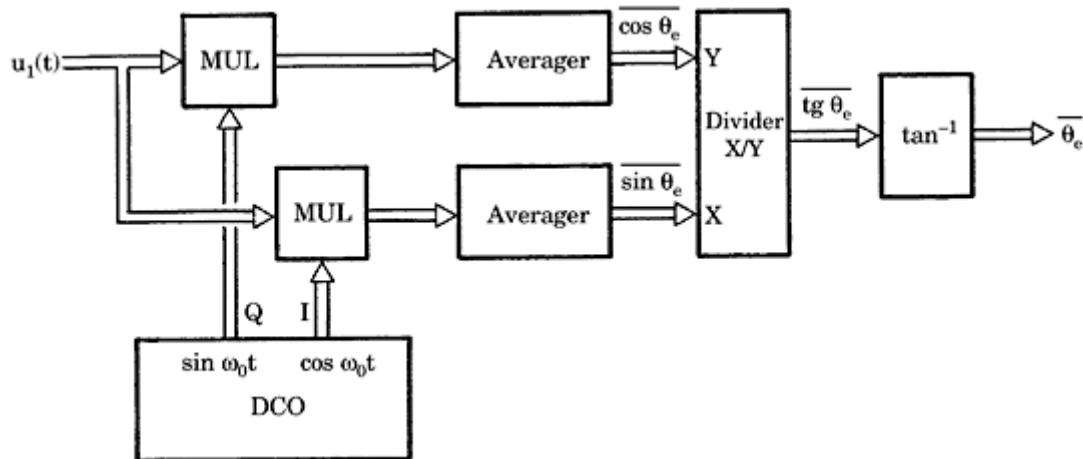


**Figure 11.4** A Hilbert-transform PD. (a) A block diagram. (b) Corresponding waveforms.

A similar but simpler way to calculate the phase error is given by the digital-averaging phase detector (see Fig. 11.5).

As in the method discussed previously, the DCO is also required to generate in-phase and quadrature signals  $I$  and  $Q$ , respectively. These are again multiplied by the digital reference

signal  $u_1(t)$ , but the signals  $\cos \theta_e$  and  $\sin \theta_e$  are obtained by simply averaging (or integrating) the output signals of the multipliers over an appropriate period of time.<sup>29</sup> Note that this arrangement already includes a filtering function, defined by the impulse transfer function



**Figure 11.5** A digital-averaging PD.

of the averaging filter used.<sup>14</sup> This method, too, lends itself particularly to implementation by software.

### All-digital loop filters

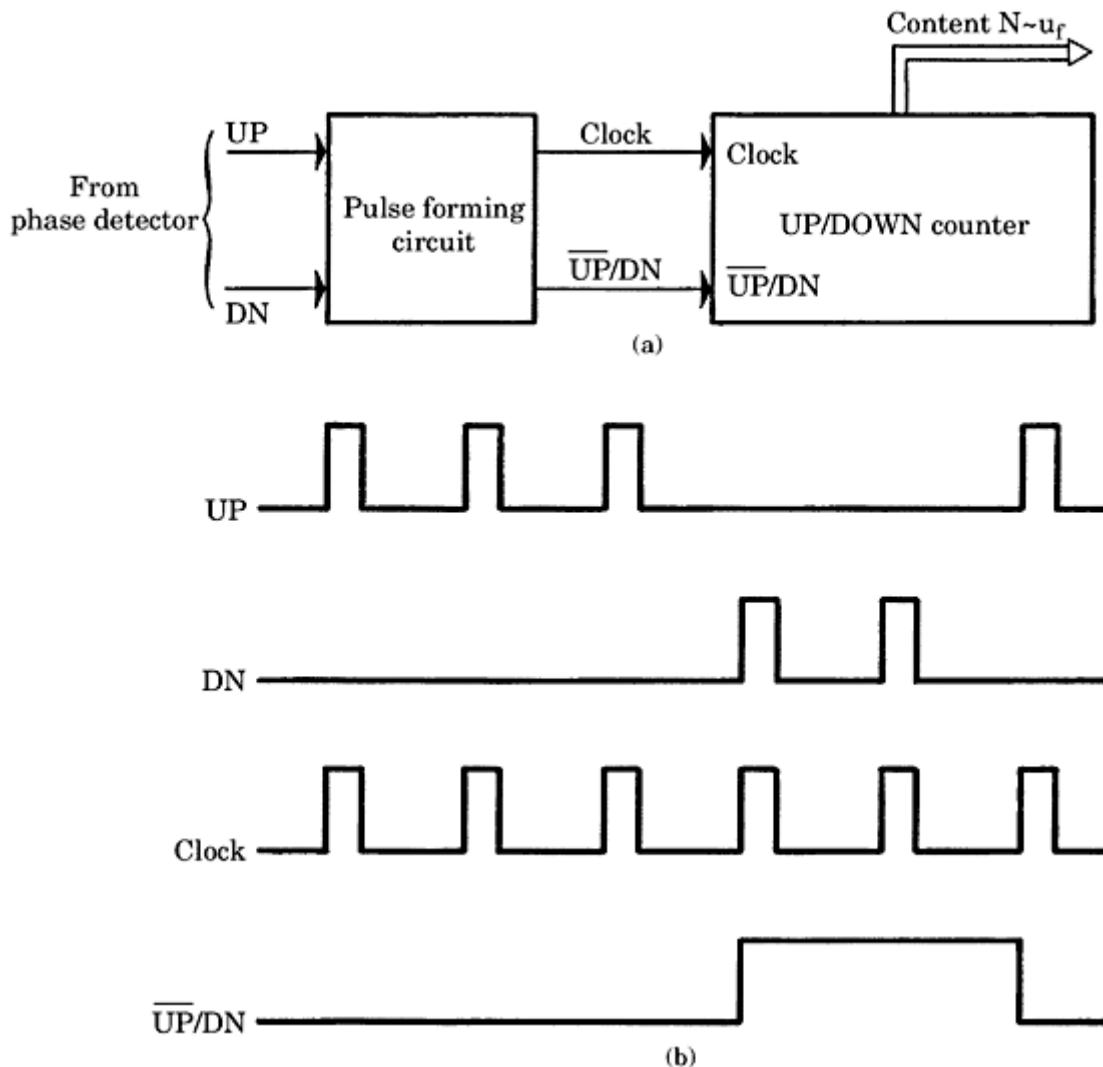
As seen in the preceding section, different all-digital PDs generate different types of output signals. The PDs discussed at the end of Sec. 11.1.1 produce  $N$ -bit digital output signals, whereas simpler types, such as the EXOR or the phase/frequency detector deliver one or two binary-valued output signals (or a tri-state signal). It becomes evident that not every all-digital loop filter is compatible with all types of all-digital PDs. Thus, we must consider which types of loop filters can be matched to the various PDs discussed previously. Probably the simplest loop filter is built from an ordinary UP/DOWN counter (Fig. 11.6a). The UP/DOWN counter loop filter preferably operates in combination with a phase detector delivering UP or DN (DOWN) pulses, such as the PFD. It is easily adapted, however, to operate in conjunction with the EXOR or JK-flipflop phase detectors and others. As shown in Fig. 11.6a, a pulse-forming network is first needed, which converts the incoming UP and DN pulses into a counting clock and a direction ( $\overline{UP/DN}$ ) signal (as explained by the waveforms in Fig. 11.6b).

On each UP pulse generated by the phase detector, the content  $N$  of the UP/DOWN counter is incremented by 1. A DOWN pulse will decrement  $N$  in the same manner. The content  $N$  is given by the  $N$ -bit parallel output signal  $u_f$  of the loop filter. Because the content  $N$  is the weighted sum of the UP and DN pulses—the UP pulses have an assigned weight of +1, and the DN pulses, -1—this filter can roughly be considered an *integrator* having the transfer function

$$H(s) = \frac{1}{sT_i} \quad (11.4)$$

where  $T_i$  is the integrator time constant. This is, however, a very crude approximation, since the UP and DN pulses do not carry any information (in this

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

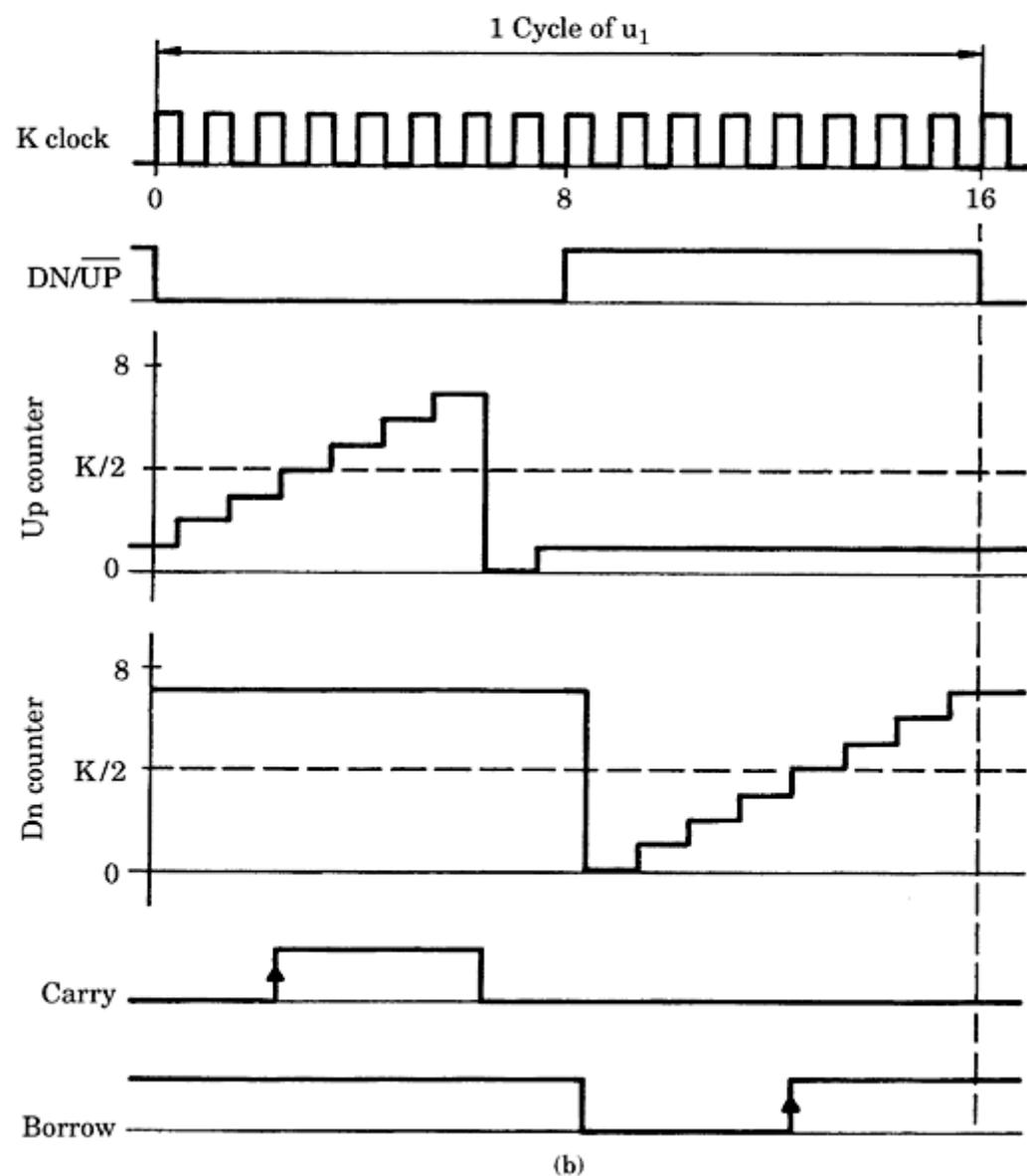
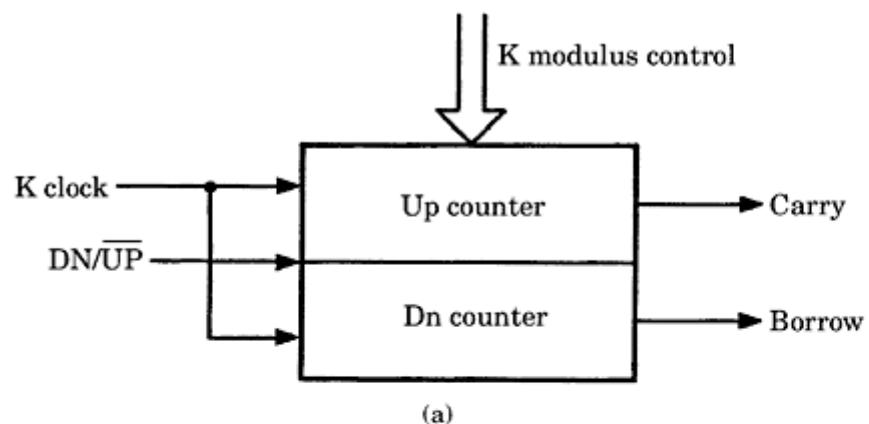


**Figure 11.6** The UP/DOWN counter loop filter. (a) A block diagram. (b) Corresponding waveforms.

application at least) about the actual size of the phase error; they only tell whether the phase of  $u_1$  is leading or lagging  $u_2$ .

One of the most important digital loop filters is the  $K$  counter (Fig. 11.7). This loop filter always works together with the EXOR or the JK-flipflop phase detector. As Fig. 11.7a shows, the  $K$  counter consists of two independent counters, which are usually referred to as “UP counter” and “DOWN counter.” In reality, however, both counters are always counting upward.  $K$  is the modulus of both counters—that is, the contents of both counters is in a range from 0 to  $K - 1$ .  $K$  can be controlled by the  $K$  modulus control input and is always an integer power of 2. The frequency of the clock signal ( $K$  clock) is by definition  $M$  times the center frequency  $f_0$  of the ADPLL, where  $M$  is typically 8, 16, 32, and so on. The operation of the  $K$  counter is controlled by the  $DN/\overline{UP}$  signal. If this signal is high, the “DOWN counter” is active, while the contents of the UP counter stays frozen. In the opposite case, the “UP counter” counts up but the DN counter stays frozen.

Both counters recycle to 0 when the contents exceed  $K - 1$ . The most significant bit of the “UP counter” is used as a “carry” output, and the most significant



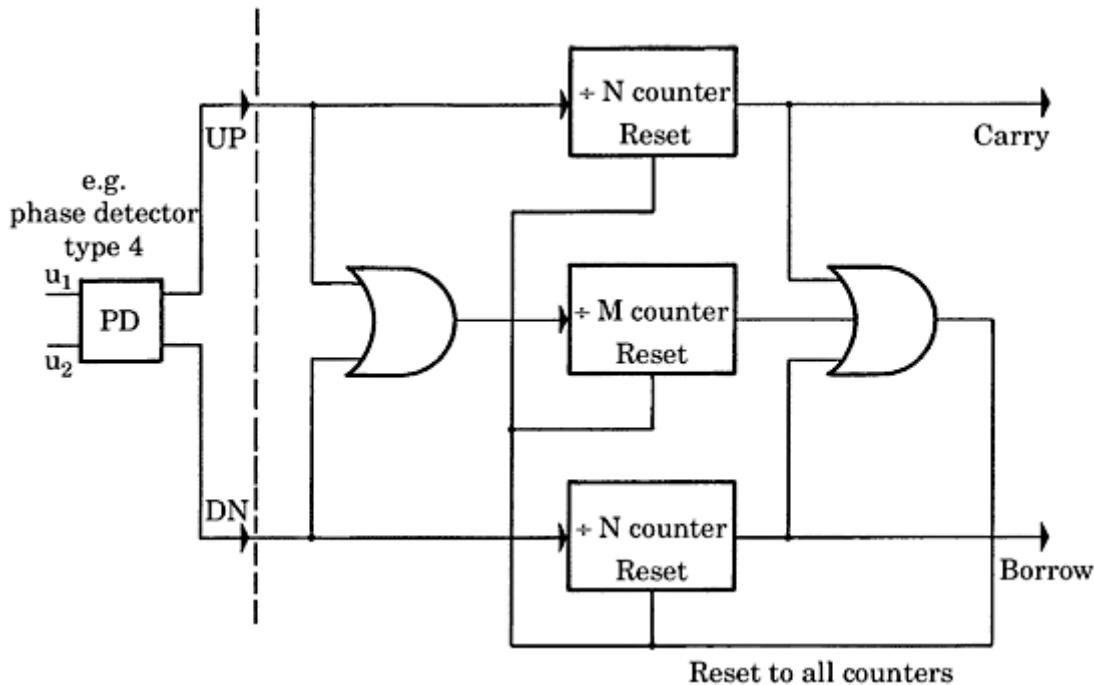
**Figure 11.7** AK counter loop filter. (a) A block diagram. (b) Corresponding waveforms.

bit of the “DN counter” is used as a “borrow” output. Consequently, the carry is high when the contents of the UP counter are equal to or more than  $K/2$ . In analogy, the borrow output becomes high when the contents of the DN counter are equal to or more than  $K/2$ . As will be shown in [Sec. 11.3](#), the positive-going

edges of the carry and borrow signals are used to control the frequency of a digital-controlled oscillator (DCO).

[Figure 11.7b](#) shows the signals of the  $K$  counter. The  $DN/\overline{UP}$  input is controlled by the output of a phase detector. In this example, it was assumed that a JK-flipflop is used and that the ADPLL operates on its center frequency. As explained by [Fig. 2.9](#), input signal  $u_1$  and output signal  $u_2'$  of the PLL are in antiphase then, and the output signal  $u_d$  of the phase detector is a square wave having an exactly 50 percent duty cycle. Hence, the  $DN/\overline{UP}$  signal is high in one half-cycle of the  $u_1$  signal and low in the other. The frequency of the  $K$  clock is assumed to be 16 times the center frequency ( $M = 16$ ). The counter modulus  $K$  has been arbitrarily set to 8. Looking at the waveforms in [Fig. 11.7b](#), we see that the UP counter counts on the first eight  $K$  clock pulses, and the DN counter counts on the next eight pulses. Under these conditions, the UP counter generates exactly one carry pulse within each cycle of the  $u_1$  signal, and the DN counter generates exactly one borrow pulse in the same interval. As will be seen later, the carry and borrow pulses then cancel. We assume now that there exists a phase error in the loop; thus, the duty cycle of the  $DN/\overline{UP}$  signal becomes asymmetric. When this signal is low during a longer fraction of one  $u_1$  cycle than it is high, the UP counter gets more clock pulses on average than the DN counter. The average number of carries then becomes larger than the average number of borrows per unit of time. When the  $DN/\overline{UP}$  signal is permanently low, the UP counter is active all the time. When the  $DN/\overline{UP}$  signal is permanently high, however, the DN counter is working continually. The  $K$  counter is part of the popular type 74HCT297 ADPLL, which will be treated in [Sec. 11.3](#).

Another digital loop filter is the so-called  $N$ -before- $M$ -counter ([Fig. 11.8](#)). The performance of this filter is very nonlinear. In [Fig. 11.8](#), it is suggested that the



**Figure 11.8** A block diagram of the  $N$ -before- $M$  loop filter.

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

$N$ -before- $M$  filter operates in conjunction with a phase detector generating UP and DN pulses, as was the case with the PFD. The  $N$ -before- $M$  filter uses two frequency counters scaling down the input signal by a factor  $N$  and one counter scaling down by  $M$ , where  $M > N$  always. The  $\div M$  counter counts the incoming UP and DN pulses, as shown in Fig. 11.8. As also seen in the diagram, the upper  $\div N$  counter will produce one CARRY output when it has received  $N$  UP pulses. But it will generate this CARRY only when the  $\div M$  counter does not receive  $M$  pulses. Otherwise, the  $\div N$  counter would have been *reset*. We can say that the upper  $\div N$  counter will produce a CARRY pulse whenever more than  $N$  pulses of an ensemble of  $M$  pulses have been UP pulses. A similar statement can be made for the lower  $\div N$  counter in Fig. 11.8, which will output BORROW pulses only when the majority of incoming pulses are DN pulses.

The outputs of the  $N$ -before- $M$  filter can be used in a similar way to control a DCO, as indicated for the  $K$  counter.

We will now deal with digital loop filters compatible with an  $N$ -bit parallel input signal. The obvious solution for this case is the *digital filter*, which operates by itself with  $N$ -bit input and  $N$ -bit output signals. With digital loop filters, any desired transfer function performed by an analog loop filter (and many additional ones) can be realized.

As we know, the performance of an analog loop filter is described by its transfer function  $F(s)$

$$F(s) = \frac{U_f(s)}{U_d(s)} \quad (11.5)$$

which is the ratio of the Laplace transforms of the signals  $u_f$  and  $u_d$  (for signal definitions, refer to Fig. 2.1). When a digital filter is realized, the transfer function  $F(s)$  of a prototype analog filter is transformed into  $z$ -domain, yielding the so-called  $z$ -transfer function  $F(z)$ , where  $z$  is the  $z$ -operator. An introduction to digital filtering is given in App. C.  $F(z)$  is the ratio of the  $z$ -transforms of the signals  $u_f$  and  $u_d$ —that is,

$$F(z) = \frac{U_f(z)}{U_d(z)} \quad (11.6)$$

When implementing the digital filter, this equation is transformed back into time domain, which yields a recursion of the form<sup>12,13,14,19</sup>

$$\begin{aligned} u_f(nT) &= b_0 u_d(nT) + b_1 u_d([n - 1]T) + b_2 u_d([n - 2]) + \dots \\ &\quad - a_1 u_f([n - 1]T) - a_2 u_f([n - 2]T) - \dots \end{aligned} \quad (11.7)$$

The signals  $u_f$  and  $u_d$  are sampled signals now, which means that they exist only at discrete time instants  $t = 0, T, 2T, \dots, nT$ , where  $T$  is the sampling interval. The  $a_i$  and  $b_i$  terms are called *filter coefficients*.

**Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.**

The sampling frequency  $f_s = 1/T$  must be chosen to be much larger than the 3-dB corner frequency of the filter, typically 10 to 20 times the corner frequency.<sup>13,14</sup>  $u_f(nT)$ ,  $u_f([n - 1]T)$  ... denote the values of the sampled signal  $u_f$  at sampling instants  $t = nT$ ,  $(n - 1)T$  ... The recursion formula calculates the output signal  $u_f(nT)$  in the  $n$ th sampling instant from the value of  $u_d$  sampled at this instant and from one or more previously sampled values of  $u_d$ .

Furthermore, in a recursive digital filter,  $u_f(nT)$  depends on values of  $u_f$  calculated in previous sampling instants. The number of “delayed” samples of  $u_f$  and  $u_d$  that have to be taken into account is equal to the order of the digital filter. (For a first-order digital filter, for example, only the filter coefficients  $a_1$ ,  $b_0$ , and  $b_1$  do not vanish.)

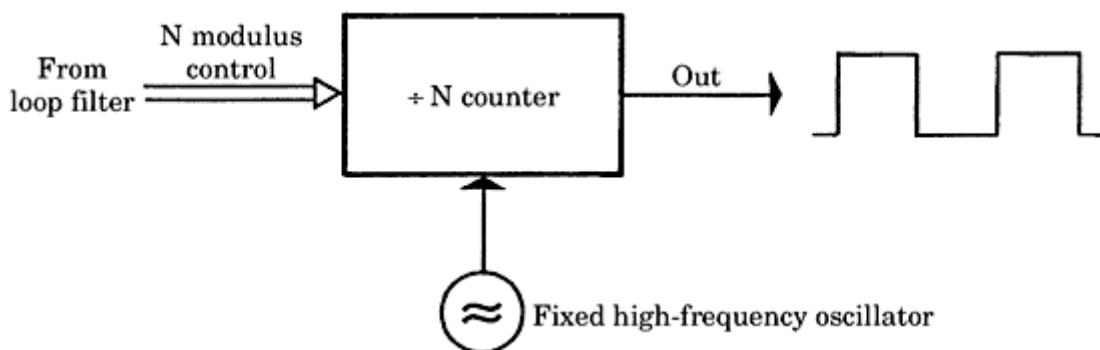
## Digital-controlled oscillators

A variety of DCOs can be designed, and they can be implemented by hardware or by software. We shall consider the most obvious solutions in this section.

Probably the simplest solution is the  $\div N$  counter DCO (Fig. 11.9). A  $\div N$  counter is used to scale down the signal generated by a high-frequency oscillator operating at a fixed frequency. The  $N$ -bit parallel output signal of a digital loop filter is used to control the scaling factor  $N$  of the  $\div N$  counter.

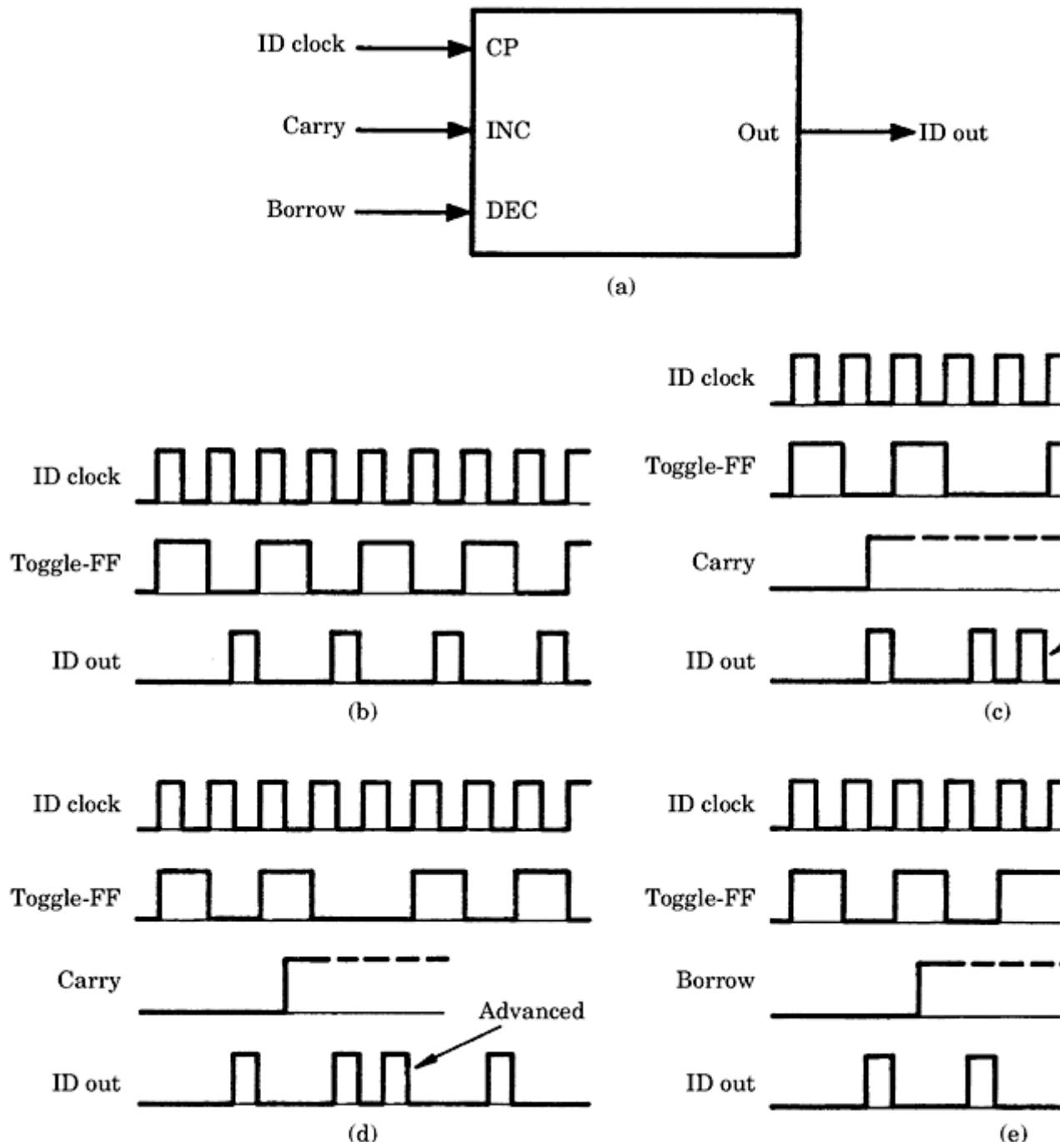
Another DCO type is the so-called *increment-decrement* (ID) counter shown in Fig. 11.10a.<sup>9,16</sup> This DCO is intended to operate in conjunction with those loop filters that generate CARRY and BORROW pulses, such as the  $K$  counter or the  $N$ -before- $M$  filter discussed in Sec. 11.1.2. The operation of the ID counter follows from the waveforms shown in Fig. 11.10b. As Fig. 11.10a shows, the ID counter has three inputs, a clock input (ID clock), an increment (INC), and a decrement (DEC) input.

Carry pulses (as delivered, for example, by a  $K$  counter; see Fig. 11.7) are fed to the INC, and borrow pulses to the DEC input. The ID counter is sensitive on the positive-going edges of the carry and borrow inputs; the duration of these signals is not of concern here. In the absence of carry and borrow pulses, the ID counter simply divides the ID clock frequency by 2; it produces an output pulse (IDout) on every second ID clock (see the waveforms in Fig. 11.10b).



**Figure 11.9** A block diagram of a  $\div N$  counter DCO.

**Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.**



**Figure 11.10** The ID counter DCO. (a) A block diagram. (b) Waveforms for the case where no carry/borrows are applied to the INC and DEC inputs, respectively. (c) Waveforms for the case where a borrow input is applied when the toggle flipflop is in the 0 state. (d) Waveforms for the case where a borrow input is applied when the toggle flipflop is in the 1 state. (e) Waveforms for the case where a borrow input is applied to the DEC input.

To understand the function of the ID counter, one must know that this circuit contains a toggle flipflop, which is not shown in the schematic diagram of Fig. 11.10a. As the waveform “Toggle-FF” in Fig. 11.10b shows, the toggle flipflop toggles on every positive edge of the

ID clock if no carries and borrows are present. The output of the ID counter (IDout) is obtained by the logical function  $IDout = \overline{IDclock} \cdot \overline{Toggle - FF}$ . Now we assume that a carry pulse appears at the INC input of the ID counter. The carry signal is processed only in the

period where the toggle flipflop is set “high.” If the carry gets “true” when the toggle flipflop is in the “low” state ([Fig. 11.10c](#)), the toggle goes high onto the next positive edge of the ID clock. It stays low, however, during two ID clock intervals thereafter. This means that the next IDout pulse is advanced in time by one ID clock period. If the carry becomes “true” when the toggle flipflop is set “high,” this flipflop is set “low” onto the next ID clock and remains low during two ID clock periods, as shown in [Fig. 11.10d](#). Because the carry can only be processed when the toggle flipflop is in the high state, the maximum frequency of the IDout signal is reached when the toggle flipflop follows the pattern “high low low high low low...” Consequently, the output frequency of the ID counter cannot be as high as the frequency of the ID clock, only (at most) two-thirds of that value. This, of course, limits the hold range of the ADPLL, as will be shown in [Sec. 11.3](#).

[Fig. 11.10e](#) demonstrates what happens when a borrow pulse is generated. In analogy, a borrow is processed only when the toggle flipflop is in the low state. As soon as a borrow is sensed, the toggle flipflop is set high onto the next positive edge of the ID clock and remains high during two ID clock periods. The next IDout pulse is therefore delayed by one ID clock period. The ID counter delivers the minimum output frequency when the toggle flipflop shows the pattern “low high high low high high...” Thus, the minimum output frequency of the ID counter is one-third the ID clock frequency. As we will see in [Sec. 11.3](#), the limited output frequency range of the ID counter restricts the realizable hold range of the ADPLL. (Note that the explanation of the ID counter performance has been slightly simplified; the actual ID counter circuit not only consists of the mentioned toggle flipflop, but also contains eight more flipflops and a number of gates. The exact operation of the ID counter can be deduced from the data sheet of the 74HC/HCT297.) Because the ID counter needs three ID clock periods to process one carry or one borrow, the maximum frequency of carry or borrow pulses must not be higher than one-third the frequency of the ID clock. If more carries or borrows are delivered, some are “overslept.” When the average frequency of the carries is such that all are processed, the instantaneous frequency of the IDout signal is increased by  $n/2$  Hz when  $n$  carries are detected in one second. This is most easily understood if we assume that the frequency of the ID clock is 32 Hz, for example.

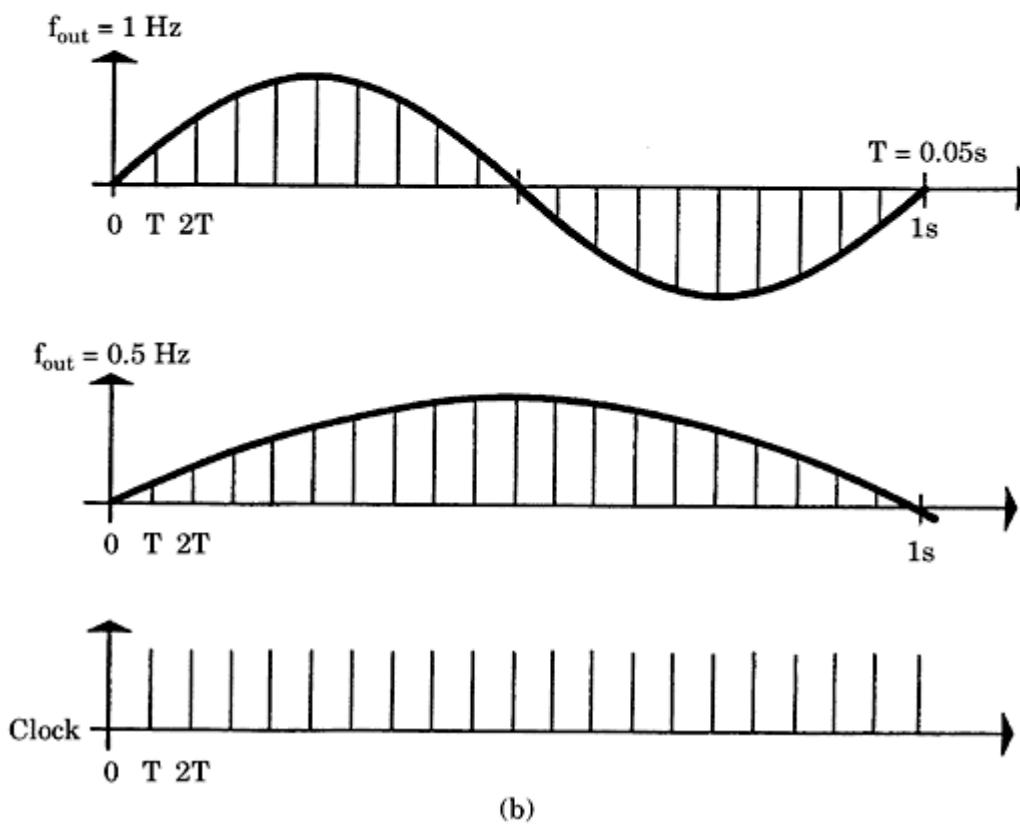
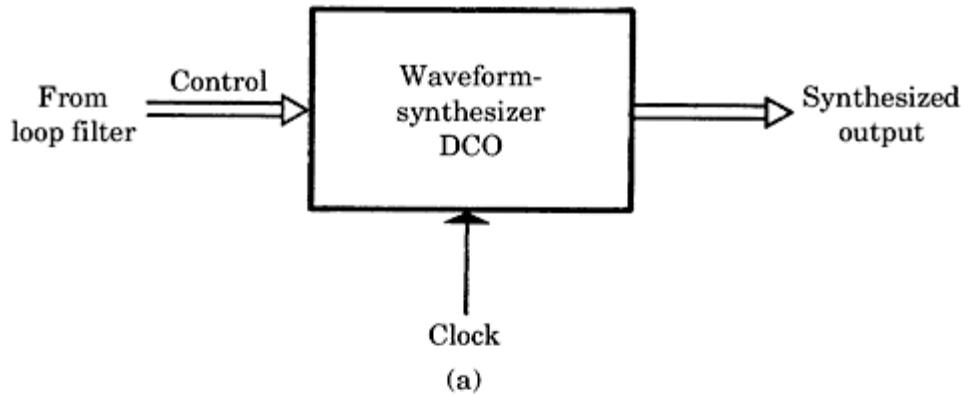
Without any carry, the output frequency would be 16 Hz. If eight carries are detected within one second, the “next” IDout pulse is advanced eight times in one second by  $1/32$  of a second. The number of output pulses is therefore increased from 16 to 20 Hz during that period, and not from 16 to 24 Hz. Generally, one carry pulse causes a  $1/2$  cycle to be added to the IDout signal, and one borrow pulse causes a  $1/2$  cycle to be deleted correspondingly.

The two DCO circuits discussed hitherto are better suited for hardware than for software implementations. The waveform synthesizer DCO—the third and last DCO to be considered here—lends itself almost ideally to implementation by software. We will discuss software implementations of the PLL in [Chap. 13](#). This type of DCO generates sine and/or cosine waveforms by looking up tables

stored in read only memory (ROM).<sup>30</sup> The block diagram of a waveform-synthesizer DCO is shown in Fig. 11.11a.

The waveforms in Fig. 11.11b demonstrate how the synthesizer generates sine waves of different frequencies (1 Hz and 0.5 Hz in this example). It operates at a fixed clock rate (that is, it calculates a sample of the synthesized signal at the sampling instants  $t = 0, T, 2T, \dots, nT$ , irrespective of the desired frequency). Lower-frequency signals are generated with higher resolution than higher-frequency signals.

In the example of Fig. 11.11b, it has been arbitrarily assumed that the sampling period is 50  $\mu\text{s}$ ; most actual waveform synthesizers operate much faster,



**Figure 11.11** A waveform-synthesizer DCO. (a) A block diagram. (b) The waveforms show

how sine waves with frequencies of 1 Hz and 0.5 Hz, respectively, are synthesized.

---

**Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.**

of course. It shows that a 1 Hz sine wave is generated with a resolution of 20 samples for a full period. In the case of the 0.5 Hz sine wave, twice as many samples (40) are produced within a full period. When generating a 1 Hz sine wave, the synthesizer calculates the sine function for the phase angles  $\Phi = 0$  (initial value),  $2\pi/20$ ,  $2 \cdot (2\pi/20)$ ,  $3 \cdot (2\pi/20)$ , and so on, and the phase angle  $\Phi$  is incremented by an amount  $\Delta\Phi = 2\pi/20$  at every clock period.

If an arbitrary frequency  $f$  is to be produced, the increment  $\Delta\Phi$  is given by

$$\Delta\Phi = 2\pi f T \quad (11.8)$$

where  $T$  is the sampling period.

As shown in [Fig. 11.11a](#), the loop filter preceding the waveform synthesizer must deliver the digital signal  $f$  (this signal is labeled “control” on the left of the figure). It is no problem to generate “simultaneously” a sine and a cosine function, as required, when a Hilbert-transform phase detector is used with an ADPLL system (refer to [Fig. 11.4a](#)).

Digital waveform synthesizers are easily implemented using single-chip microcomputers.<sup>30</sup> The speed of trigonometric computations can be greatly enhanced by using table-lookup techniques rather than by calculating a sine function with a Taylor series or Chebyshev polynomials.

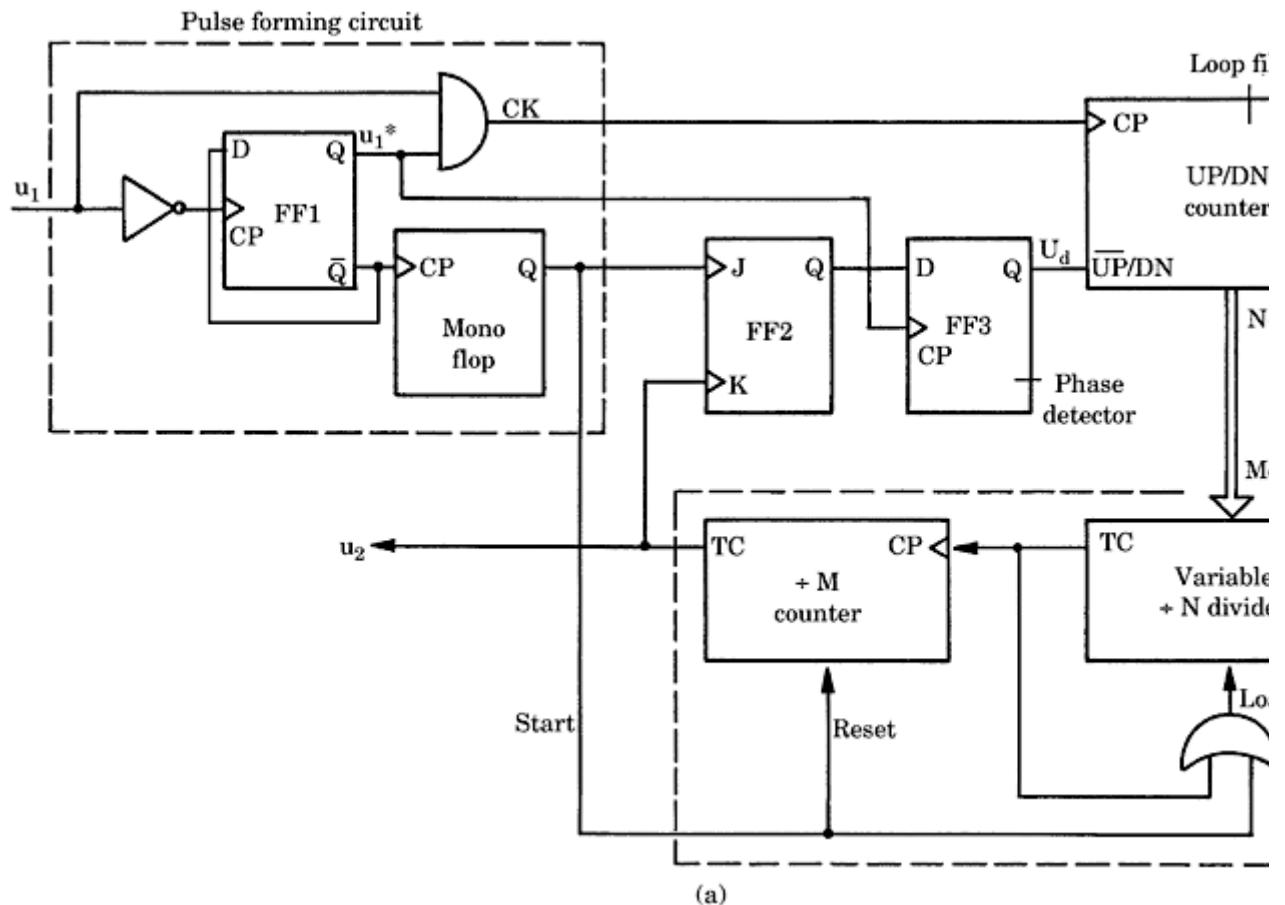
An example of the application of a waveform-synthesizer DCO is presented in [Sec. 11.2](#).

## Examples of Implemented ADPLLs

Based on the numerous variants of all-digital PDs, loop filters, and DCOs, an almost unlimited number of ADPLLs can be built by combining compatible functional blocks. There is extended literature on this subject, and a review of the most important systems can be found in [ref. 8](#). A detailed discussion of every possible ADPLL system would go beyond the scope of this book; we therefore consider only three typical ADPLL implementations.

The first two are hardware implementations. There is no reason why they could not be designed using software as well. The last example to be discussed is a typical software-based system, which encompasses a large variety of mathematical operations. A hardware implementation of this type of PLL is certainly not impossible, but the hardware would be very complex.

The first example of an ADPLL is depicted in [Fig. 11.12a](#).<sup>39</sup> In this circuit, the input signal  $u_1$  is first preprocessed by a pulse-forming network (cf. dashed box on the left). The generated signals are shown in [Fig. 11.12b](#). First, D-flipflop FF1 scales down the frequency of the input signal by a factor of 2. The down-scaled signal is denoted  $u_1^*$ . By AND-ing  $u_1$  and  $u_1^*$ , a clock signal CK is generated that is applied to the counting input of an UP/DOWN counter. The signal  $u_1^*$  is used to set the state of D-flipflop FF3, which serves as a phase detector. The duration of CK is one-quarter of the period of  $u_1^*$ . Moreover, the negative going edges of  $u_1^*$  trigger a monoflop which generates very short pulses. These pulses are labeled *Start*. Their duration must be shorter than the period of the high-fre-



**Figure 11.12** An all-digital PLL system, example 1. (a) A block diagram. (b) Corresponding waveforms. Two cases are shown: (1) a divider ratio  $N$  too small; (2) a divider ratio  $N$  too large.

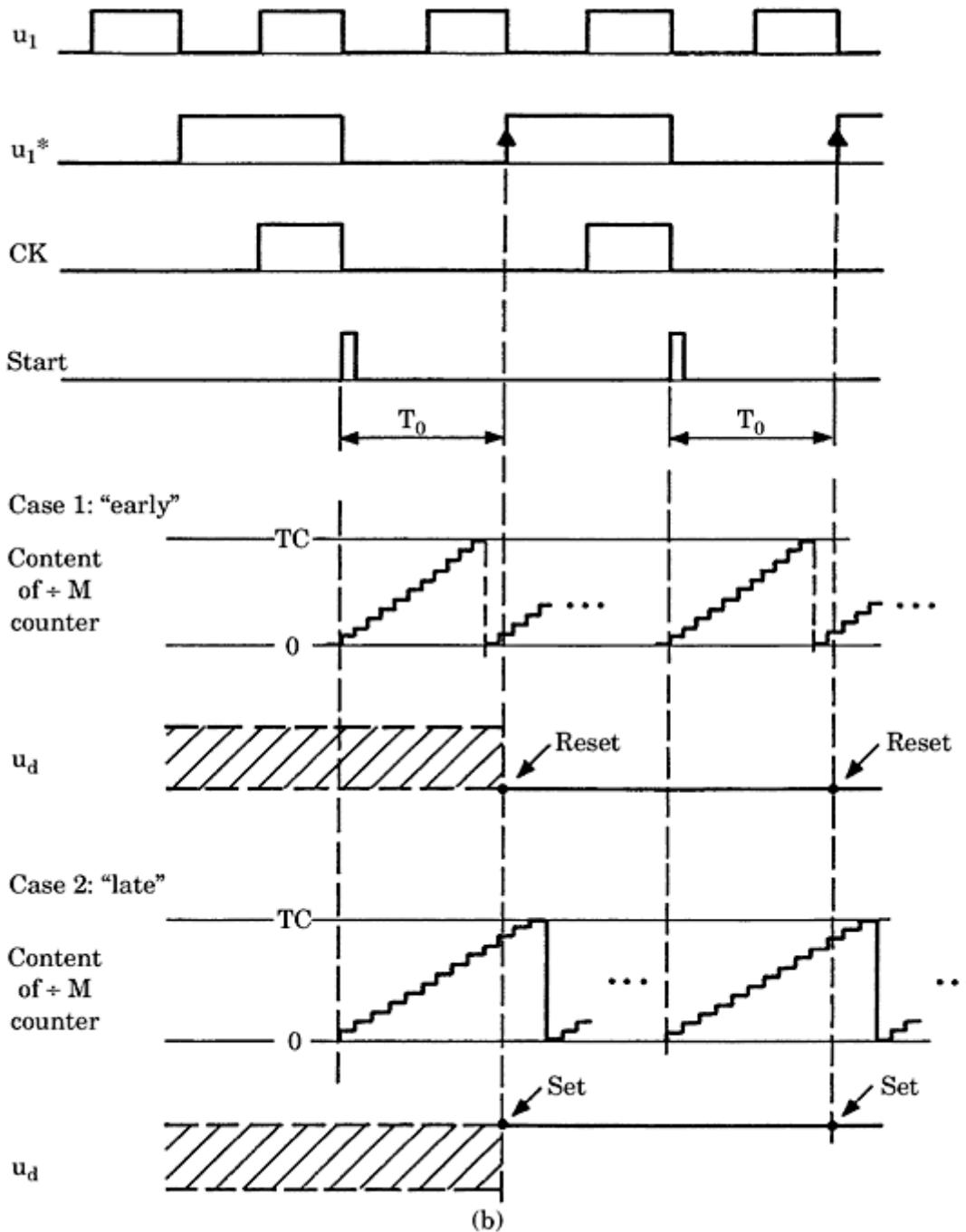
quency clock  $f_c$  which will be discussed in the following. The dashed enclosure on the bottom right of Fig. 11.12a represents a DCO (digital-controlled oscillator). It is built from a cascade of two counters, a variable modulo- $N$  divider and a fixed modulo- $M$  counter.

The variable modulo- $N$  divider is a DOWN counter. Its content starts with the number  $N$ , which is loaded in parallel by the *Load* input. The clock signal  $f_c$  causes the divider to count down. When it reaches the terminal count (TC)—which is 0 in this case—a pulse is delivered at the TC output. This immediately reloads the content  $N$  (see the OR gate at the *Load* input), and the modulo- $N$  divider continues counting down. The fixed modulo- $M$  counter is an UP counter. Its content is reset to 0 upon applying a *Reset* signal. The pulses applied to the CP input ramp up its content until it reaches the terminal count, which is a positive number here. As soon as the terminal count is reached, a pulse is delivered at the TC output, and the content wraps around to zero again. The counter then continues counting up.

When the circuit is locked, the clock frequency  $f_c$  should be

$$f_c = NMf_1 \quad (11.9)$$

where  $f_1$  is the frequency of the input signal  $u_1$ . If this condition is met, the high-frequency clock delivers exactly  $N \cdot M$  pulses during one cycle of the input

**Figure 11.12 (continued)**

signal  $u_1$ , whose period is marked  $T_0$  in Fig. 11.12b. In this case, the frequency of  $u_2$  equals the frequency of  $u_1^*$ —that is,  $f_1/2$ . When the locking process starts, the modulus  $N$  of the variable modulo- $N$  divider can have an arbitrary value; thus,  $N$  is either too high or too low. The phase detector must adjust the value of  $N$  by increasing or decreasing the content of the UP/DN counter, which is shown at the top right of Fig. 11.12a, until  $N$  has the correct value. Because the UP/DN counter immediately controls the frequency of the DCO, this counter acts

as a loop filter. To see how the loop becomes locked, we want to check the waveforms in [Fig. 11.12b](#).

The START pulse resets the modulo- $M$  counter on each high-to-low transition of signal  $u_1^*$  (see the waveform “Content of  $\div M$  Counter”). As mentioned earlier, the duration of the START pulse must be shorter than the period of the  $f_c$  clock. If its pulse width were chosen larger, the LOAD pulse of the modulo- $N$  divider would last during several cycles of the  $f_c$  clock, thus inhibiting the counter to change its content. If  $N$  already had its correct value, the terminal count of the modulo- $M$  counter would be reached exactly  $T_0$  seconds after reset. In a first case named “*Case 1: early*,” it was assumed that  $N$  is smaller than required, thus the clock frequency at the CP of the modulo- $M$  counter is too high, and the terminal count is reached in shorter time—in other words, before the next low-to-high transition of  $u_1^*$ . The positive-edge-triggered JK-flipflop FF2 was initially set to its 1 state by the START pulse. Now the TC output of the modulo- $M$  counter resets FF2 before the low-to-high transition of  $u_1^*$  occurs. (Note that FF2 is an edge-triggered JK-flipflop.) Consequently, the next positive edge of  $u_1^*$  resets D-flipflop FF3, whose output is labeled  $u_d$  (phase-detector output) in Figs. 11.12a and 11.12b. Note that the state of  $u_d$  was indeterminate at the start of the locking process; hence, its waveform is drawn by the shaded area in Fig. 11.12b. Because  $u_d$  is low now, the next CK (clock) pulse causes the UP/DN counter to increase its content ( $N$ ) by 1. This lowers the instantaneous frequency of the TC output of the modulo- $N$  divider, and the terminal count of the modulo- $M$  counter will be reached later in the next cycle of  $u_1^*$ . This process repeats until the modulo- $M$  counter reaches TC after occurrence of the positive edge of  $u_1^*$ .

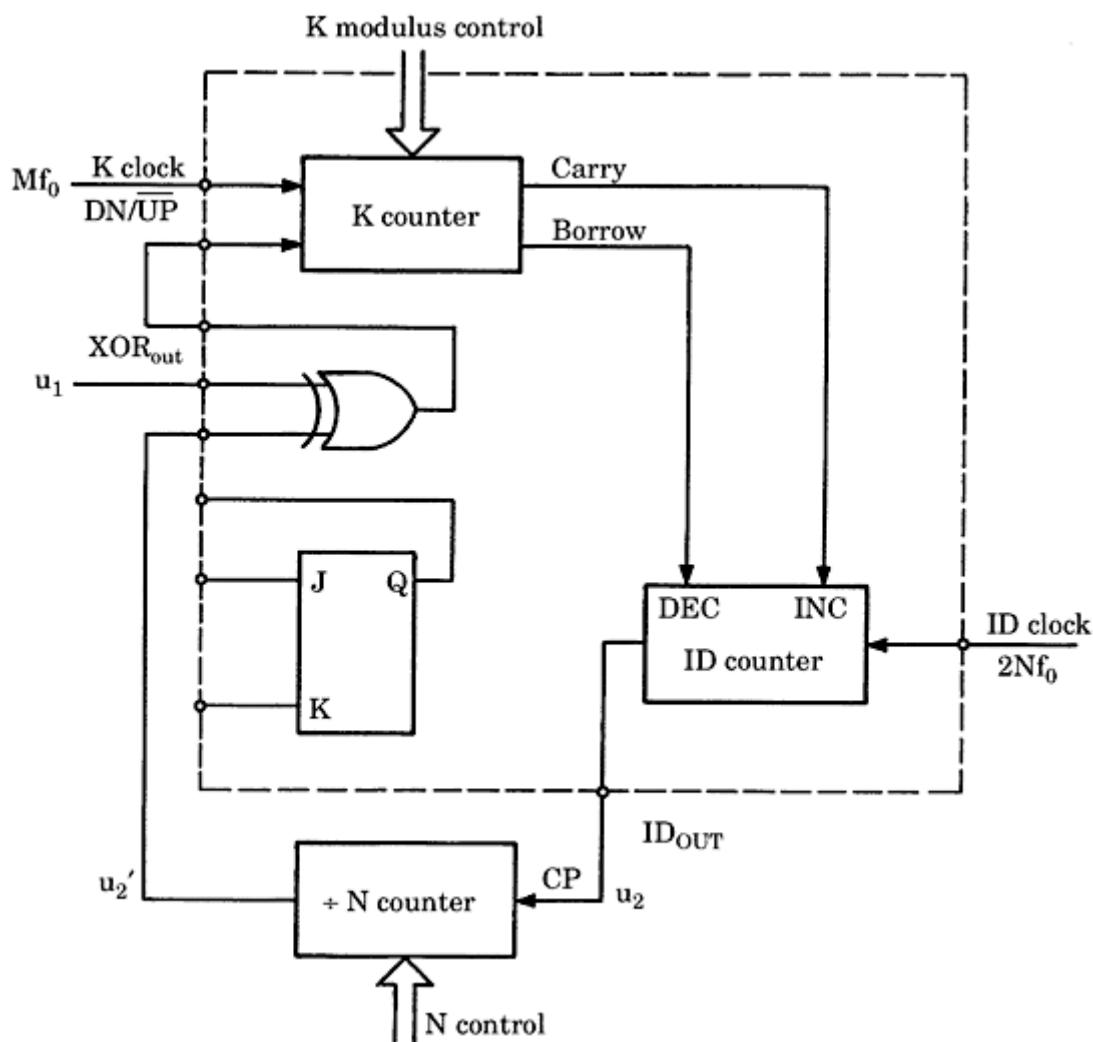
In the other case named “*Case 2: late*,”  $N$  was too large initially, thus the modulo- $M$  counter required a period longer than  $T_0$  to reach TC. Under this condition, D-flipflop FF3 will be set to 1 on the next low-to-high transition of  $u_1^*$ . This causes the next CK pulse to decrease the content  $N$  of the UP/DN counter. Consequently, the instantaneous output frequency of the modulo- $N$  divider will become higher. When the lock process has been completed, the content  $N$  of the UP/DN counter will usually toggle between two adjacent values  $N$  and  $N + 1$  in successive cycles of  $u_1^*$ . As a numerical example, assume that the high-frequency clock is  $f_c = 10$  MHz and the input frequency is  $f_1 = 10.1$  kHz. The overall divider ratio of the cascade of both counters (modulo- $M$  and modulo- $N$ ) then should be  $N \cdot M = 990.1$ . Supposing that  $M = 100$  (in other words, two cascaded decade counters are used for the modulo- $M$  counter),  $N$  will toggle between 9 and 10 in alternate cycles. This obviously results in a phase jitter of the ADPLL’s output signal  $u_2$ . The phase jitter can be made arbitrarily small by increasing the frequency  $f_c$  of the clock signal.  $N$  will then settle at higher values. Note, however, that the lock-in process will become slower then, because the UP/DN counter would probably have to change its initial content much more but can increase or decrease it only in increments of 1 in one cycle of  $u_1^*$ .

Basically, this circuit is highly nonlinear, but it turns out that its inherent stability is just a consequence of this nonlinearity. When trying to model the circuit, we become aware that the UP/DN counter that acts as a loop filter behaves like an integrator. If a phase error persists for an extended period of time, the

**Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.**

content of the UP/DN counter ramps up or down depending on the sign of the phase error, thus performing like an integrator. As we know from the theory of the PLL, a VCO is also modeled as an integrator, because its output phase  $\theta_2$  is proportional to the integral of applied control signal  $u_f$  [cf. Eq. (2.33a)]. An analogous statement can be made for a DCO; hence, our circuit contains a cascade of two integrators, which implies that its phase-transfer function  $H(s)$  has two poles at  $s = 0$ . Because there are no compensating “zeroes” in this system, it would become unstable. Most happily, this does not occur, because the contents of the counters within the DCO are reset on every START pulse, as described earlier. In other words, the “integral” term at the output of the DCO is not allowed to “wind up.” Due to the nonlinearities of the circuit, it becomes very difficult to establish a mathematical model. No such model has been developed to the author’s knowledge.

The second ADPLL system described here is shown in Fig. 11.13. This is the most often used ADPLL configuration, and is available as an integrated circuit with the designation 74xx297, where xx stands for the family specification



**Figure 11.13** An all-digital PLL, example 2. This circuit is based on the familiar IC of type 74HC/HCT297. The EXOR phase detector is used here. For the external  $\div N$  counter, an IC of the type 74HC/HCT4040 can be used.

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

(HC, HCT, LS, S, and so on). We will analyze this ADPLL in greater detail in [Sec. 11.3](#). The IC contains two phase detectors: an EXOR gate and a JK-flipflop. In the schematic of [Fig. 11.13](#), the EXOR is used. The loop filter is formed by the previously discussed  $K$  counter ([Fig. 11.7](#)), and the already known ID counter ([Fig. 11.10](#)) is used as DCO. This ADPLL system requires an external divide-by- $N$  counter.

The system is supposed to operate at a center frequency  $f_0$  referred to the input  $u_1$ . The  $K$  counter and the ID counter are driven by clock signals having frequencies of  $M$  times and  $2N$  times the center frequency  $f_0$ , respectively. Normally, both  $M$  and  $2N$  are integer powers of 2 and are mostly derived from the same oscillator. In many cases,  $M = 2N$ , so both clock inputs can be tied together.

Assume for the moment that the EXOR phase detector is used and that the ADPLL operates on its center frequency. Then the ID counter is required to scale down the ID clock precisely by 2. The average number of carry and borrow pulses delivered by the  $K$  counter must therefore be the same, too. This is possible only when the phase difference between the signals  $u_1$  and  $u_2'$  is  $90^\circ$ . In this case, the output signal of the EXOR gate is a symmetrical square wave whose frequency is twice the center frequency. Consequently, the UP counter will count during two quarters of the reference cycle, and the DOWN counter will count in the remaining two quarter periods. Because the average number of carries and borrows precisely matches, no cycles are added to, or deleted from, the ID counter. If the reference frequency is increased, however, the output signal of the EXOR phase detector must become asymmetrical in order to allow the  $K$  counter to produce more carries than borrows on average.

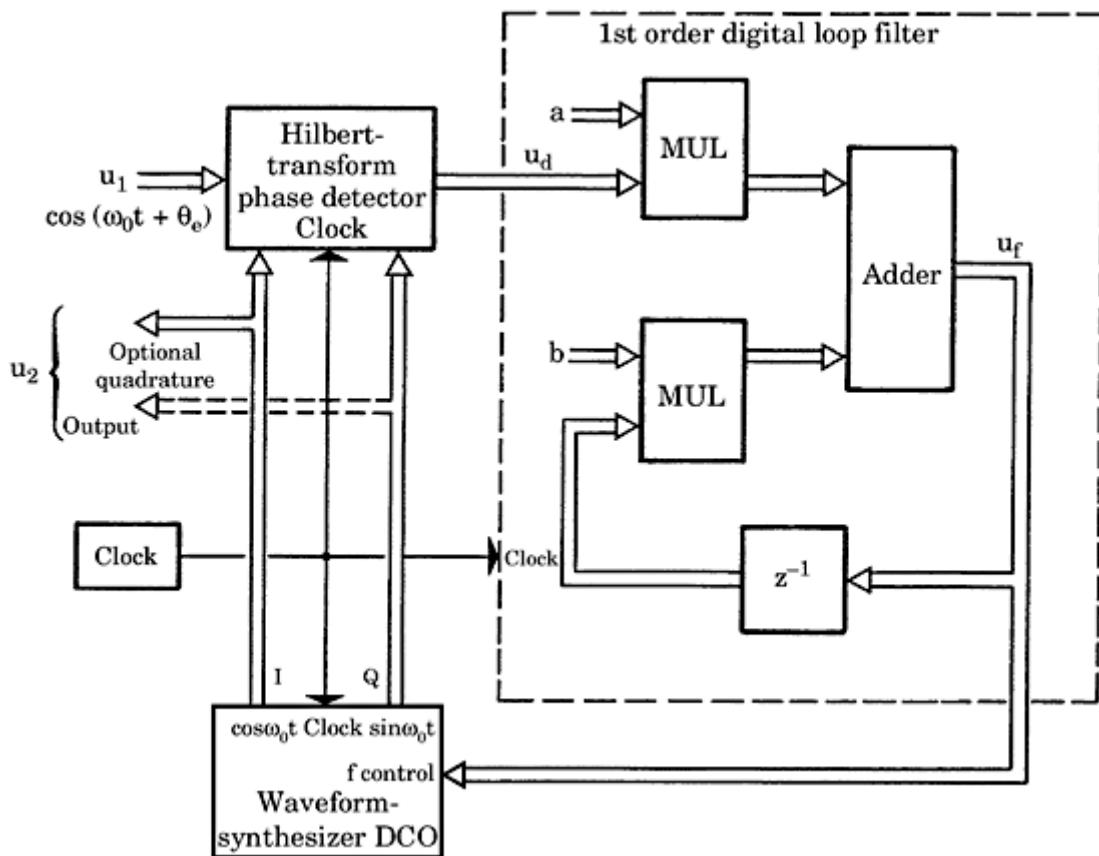
The last example of an ADPLL is illustrated in [Fig. 11.14](#). A similar system has been implemented by software on a TMS320 single-chip microcomputer (Texas Instruments).<sup>30</sup> The system of [Fig. 11.14](#) is built from functional blocks introduced previously:

- A Hilbert-transform phase detector ([Fig. 11.4](#))
- A first-order digital loop filter
- A waveform-synthesizer DCO ([Fig. 11.11](#))

As indicated in the block diagram, the arithmetic and logic operations within the functional blocks are performed under control of a clock. This means that all routines calculating the output variables of the blocks are executed periodically. The DCO generates the in-phase and quadrature signals  $I$  and  $Q$  required by the Hilbert-transform PD to calculate the phase error,  $u_d \equiv \theta_e$ . The output signal  $u_d$  is digitally filtered by the loop filter, which performs the operation

$$u_f(nT) = b_0 u_d(nT) - a_1 u_f[(n-1)T] \quad (11.10)$$

where  $a_1$  and  $b_0$  are filter coefficients, as defined in [Eq. \(11.7\)](#). The mathematical operations performed by the digital filter are represented by the dashed block in [Fig. 11.14](#).



**Figure 11.14** An all-digital PLL system, example 3. This system is best implemented by software.

One of the major benefits of the software implementation is the simplicity of changing the structure of the ADPLL system. With only minor program modifications, the first-order loop filter could be turned into a second-order one.<sup>30</sup> This would yield a third-order PLL.

## Theory of a Selected Type of ADPLL

Because there are so many variants of purely digital phase detectors, loop filters, and controlled oscillators, an enormous number of different ADPLL systems can be built. Among these variants, some will perform similarly to LPPLs. Others will operate like classical DPLLS, but the functioning of many ADPLLs will have almost nothing in common with LPPLs and DPLLS. For this reason, it is absolutely impossible to create a generalized “theory of the ADPLL.” To investigate the behavior of a particular ADPLL type, the user is forced to look for appropriate models of the corresponding function blocks and then try to get a reasonable description in the form of transfer functions (for example, phase transfer functions), Bode diagrams, or the like. In many cases, the application of standard tools (like linear control theory) will fail, because the systems to be analyzed are mostly nonlinear.

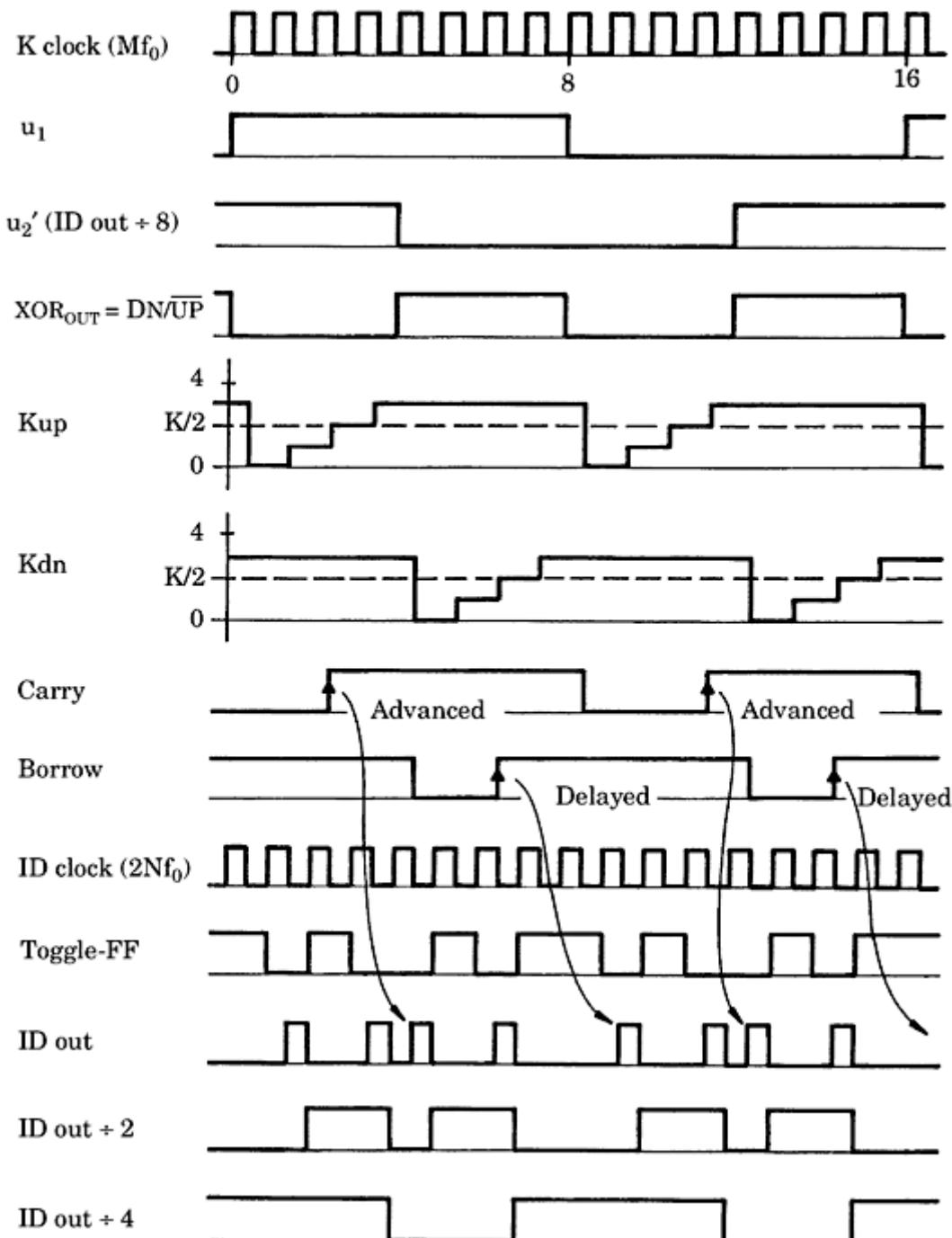
To demonstrate that analyzing an ADPLL is not an entirely hopeless job, we investigate the dynamic performance of the most popular ADPLL type, the familiar 74HCT297, which was already shown in Fig. 11.13.

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

## The effects of discrete-time operation

It is our aim to investigate the most important key parameters such as hold range, lock range, and lock-in time. We assume for the moment that the EXOR is used as a phase detector, as shown in [Fig. 11.13](#). Performance of the ADPLL is most conveniently analyzed by the waveforms of the circuit, which are shown in [Fig. 11.15](#). The signals are plotted for the simple case that the reference frequency  $f_1$  equals the center frequency  $f_0$ . The frequency of the  $K$  clock has been



**Figure 11.15** Waveforms of an ADPLL system using the IC type 74HC/HCT297. The ADPLL operates at its center frequency. The EXOR PD is used, and the parameters of the circuit are  $M = 16$ ,  $K = 4$ , and  $N = 8$ . Note that  $K$  has been chosen as small to simplify the drawing. In a real application, however,  $K$  cannot be less than 8.

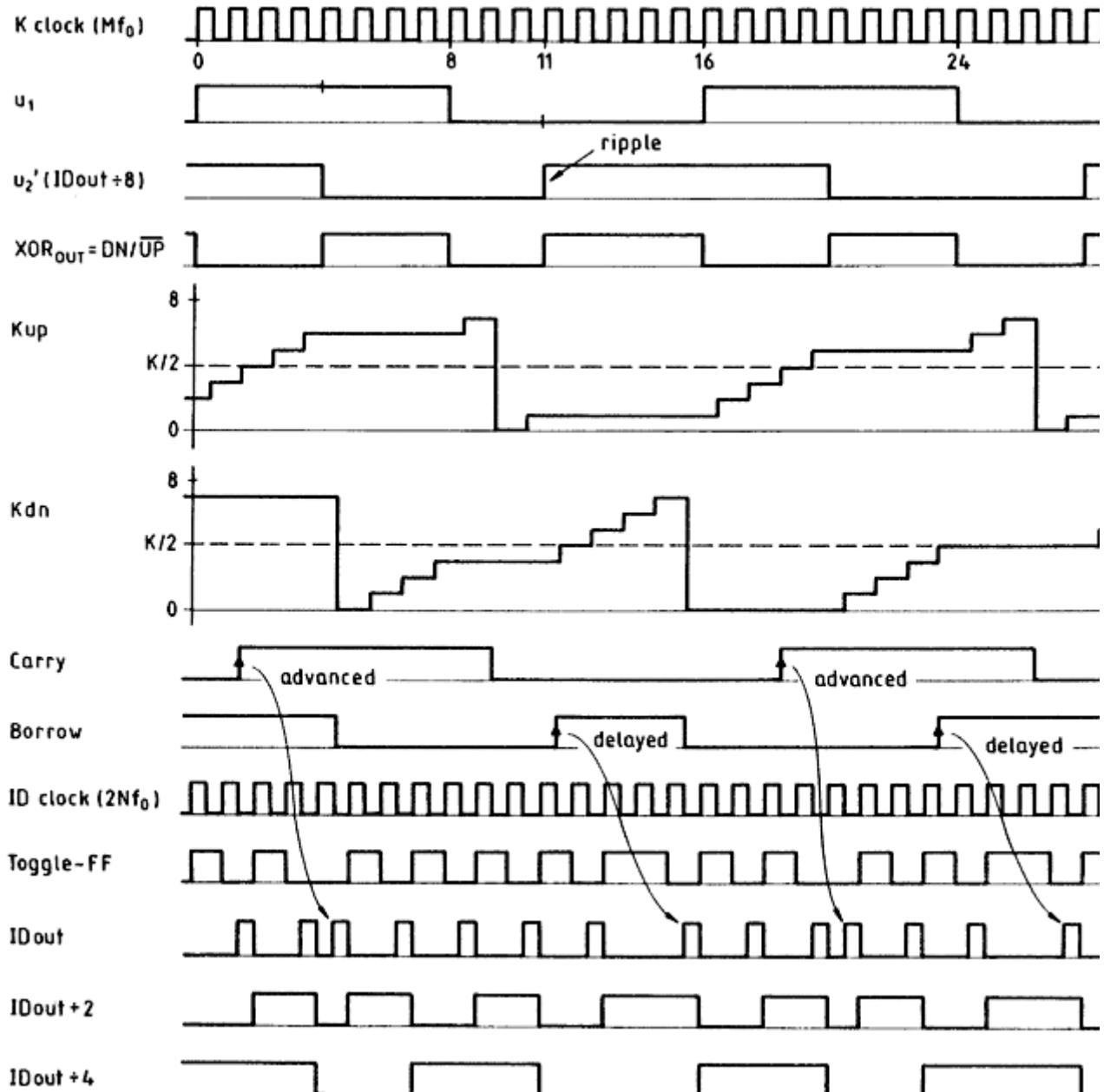
chosen to be 16 times the clock frequency ( $M = 16$ ). The  $K$  counter modulus is assumed to be 4 ( $K = 4$ ). (Note, however, that the minimum value of  $K$  for the 74HC/HCT297 is 8.) The divider ratio of the  $\div N$  counter is 8 in this example ( $N = 8$ ), so the  $K$  clock and the ID clock can be taken from the same generator. One cycle of the reference signal  $u_1$  consists therefore of 16 cycles of the  $K$  clock. The contents of the UP and DOWN counters are denoted  $K_{up}$  and  $K_{dn}$ , respectively. As can be seen from the data sheet of the 74HC/HCT297, these two counters are reset when power is applied to the circuit. When it has been operating for an undefined period of time, the contents will be arbitrary at a given time. At the instant where the 0th  $K$  clock occurs (Fig. 11.15),  $K_{up}$  and  $K_{dn}$  can therefore be assigned arbitrary numbers. For the polarities of the clock pulses, the following conventions are made:

- Both counters of the  $K$  counter count onto the negative edges of the  $K$  clock.
- The toggle flipflop within the ID counter toggles onto the positive edge of the ID clock.
- All flipflops of the  $\div N$  counter count onto the negative edge of the corresponding clock signal—in other words, the first stage of the  $\div N$  counter (denoted  $ID_{out} \div 2$  in Fig. 11.15) counts onto the negative edge of  $ID_{out}$ , the second stage (denoted  $ID_{out} \div 4$ ) counts onto the negative edge of the  $ID_{out} \div 2$  output signal, and so on.

As we see from the  $K_{up}$  and  $K_{dn}$  waveforms, the UP counter is active during the first and third quarters of the reference cycle, whereas the DOWN counter is active during the second and fourth. Hence, carries appear on ID clocks 2 and 10, while borrows are detected on ID clocks 6 and 14. (Remember that the carry and borrow signals depicted in Fig. 11.15 are simply the outputs of the most significant bit of the corresponding counter. Hence the carry becomes high when the content of the UP counter has reached  $K/2$ .) As the  $ID_{out}$  waveform shows, its pulses are periodically advanced and delayed by one cycle of the ID clock. The bottommost two signals represent the scaled-down  $ID_{out}$  signal, where the scaling factors are 2 and 4, respectively.

Because of the carry and borrow pulses, the output of the toggle flipflop becomes asymmetric. Therefore, the  $ID_{out}$  signal does not have constant frequency but rather exhibits phase jitter. The output signals  $ID_{out} \div 2$  and  $ID_{out} \div 4$  are also asymmetrical, but the output signal  $u_2'$  (which corresponds to  $ID_{out} \div 8$ ) is symmetrical again. The reason for this is that there is exactly one carry and one borrow in the period where  $u_2'$  is high, and there is exactly one carry and one borrow in the period where  $u_2'$  is low. The ripple introduced by delaying and advancing the  $ID_{out}$  pulses is therefore canceled at the output of the  $\div N$  counter.

It would be premature, however, to conclude that there is never ripple in the output signal  $u_2'$  of this type of ADPLL. Let us choose a larger value for  $K$  in the next example—for example,  $K = 8$ . All other parameters remain unchanged. Figure 11.16 shows what happens now.



**Figure 11.16** This is like Fig. 11.15, but the parameters of the ADPLL are  $M = 16$ ,  $K = 8$ , and  $N = 16$ . ADPLL operates at its center frequency. Note that the UP counter and the DOWN counter recycle only on every second UP-counting or DOWN-counting period, respectively.

When the ADPLL operates at its center frequency again, the UP counter counts up by 4 in one quarter period of the reference signal  $u_1$  on average, and the DOWN counter counts up by 4 on one quarter period of  $u_1$  on average. Carries and borrows are now generated only in each second UP-counting or DOWN-counting intervals. Figure 11.16 shows two periods of the reference signal, which corresponds to 32  $K$  clock cycles. Carries are produced onto  $K$  clocks 1 and 18, and borrows are produced onto  $K$  clocks 11 and 23. Because advancing and delaying

of the IDout pulses no longer cancel in a particular half-cycle of  $u_2'$ , this signal shows ripple; note the half-cycle of  $u_2'$ , for example, which goes from K clock 4 to 11. Its duration is seven K clock pulses instead of

eight. In succeeding reference periods (not shown in this figure), there must be half-cycles of  $u_2'$  that have a duration of nine  $K$  clock pulses. We conclude that there is no ripple on the output signal if the  $K$  modulus is chosen such that it produces exactly one carry (or one borrow) in a quarter period of the reference signal. Choosing

$$K = M/4 \quad (11.11)$$

provides zero ripple when the EXOR phase detector is used. An ADPLL having  $K = M/4$  is therefore called a “minimum ripple” configuration. As we will see later, every ADPLL exhibits some ripple if it operates at other frequencies.

If  $K > M/4$ , ripple is produced. It is easy to calculate the amount of ripple to be expected under this condition. Ideally, the duty factor  $\delta$  of the output signal  $u_2'$  is  $\delta = \delta_0 = 0.5$ . If carries and borrows in succeeding UP- and DOWN-counting periods do not cancel, the edges of the signal  $u_2'$  can be advanced or delayed by at most one ID clock cycle—in other words, by a time interval of  $1/(2Nf_0)$ . Consequently, the actual duty factor can vary in the range

$$0.5\left(1 - \frac{1}{N}\right) < \delta < 0.5\left(1 + \frac{1}{N}\right) \quad (11.12)$$

thus, the relative duty factor deviation is  $1/N$  at worst. It will be demonstrated later that ripple can be suppressed by the addition of only a few components.

Let's check what happens if  $K$  is chosen smaller than  $M/4$ . If we consider Fig. 11.15 again and assume  $K = 2$  now (which is not possible, however, with the 74HC/HCT297 IC), the UP counter would generate two carries in one UP-counting period, and the DOWN counter would generate two borrows in one DOWN-counting period. Because two carries and two borrows would cancel in succeeding UP-counting and DOWN-counting periods, there should theoretically be no ripple in the  $u_2'$  waveform. This is true only, however, when the ID clock frequency is chosen large enough so that the ID counter is able to process all carries and borrows.

When the UP counter is counting up for an extended period of time, it produces a carry every  $K/(Mf_0)$  seconds. If a number of carries have to be processed in succession by the ID counter, the delay between any two carries should be larger than three ID clock periods, as shown in Sec. 11.1.3 (see also Fig. 11.10). Because the duration of one ID clock cycle is  $1/(2Nf_0)$  seconds, we have no over-slept carries (or borrows) when the condition

$$N > N_{\min} = \frac{3M}{2K} \quad (11.13)$$

is met. Now  $M$ ,  $K$ , and  $N$  are mostly integer powers of 2, so in practical applications the minimum  $N$  will be chosen

$$N > N_{\text{pract}} = \frac{2M}{K} \quad (11.14)$$

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

where  $N_{\text{pract}}$  is also an integer power of 2. In this example, we have  $N_{\text{pract}} = 2 \cdot 16/2 = 16$ . Because we have chosen  $N = 8$ , the frequency of the carries and borrows would clearly be too high, so the ADPLL would not work properly.

Generally, we can conclude that for an ADPLL using the EXOR phase detector we have minimum ripple when  $M$  is chosen to produce at least one carry in a quarter cycle of  $u_1$  ( $K \leq M/4$ ) and when  $N$  is chosen such that all carries and borrows can be processed ( $N \geq 2M/K$ ). When  $K$  is chosen greater than  $M/4$ , less than one carry (or borrow) is generated within one quarter period of  $u_1$  on average, so ripple will be created. The amount of ripple is given by Eq. (11.12).

Now we have to investigate how the ADPLL performs when the JK-flipflop phase detector is used. We suppose that the signals  $u_1$  and  $u_2'$  are connected to the  $J$  and  $K$  inputs of the flipflop in Fig. 11.13, respectively, and that the  $Q$  output of this flipflop is tied with the  $\text{DN}/\overline{\text{UP}}$  input of the  $K$  counter. As we know from the theory of the DPLL, the signals  $u_1$  and  $u_2'$  should be in antiphase when the PLL operates at its center frequency. The waveforms for this example are shown in Fig. 11.17 for the parameters  $M = 16$ ,  $K = 8$ , and  $N = 8$ . If both signals  $u_1$  and  $u_2'$  were precisely symmetrical, the DOWN counter would be active in the first half of the reference cycle, and the DOWN counter would be active in the second half. On average, both UP and DOWN counters would overflow once in a counting period. As the waveforms show, the DOWN counter is active in the first half of the cycle. Because the borrow causes one of the IDout pulses to be delayed by one ID clock, the waveform for  $u_2'$  becomes asymmetric. The asymmetry is easily calculated in this case, too. In one upward-counting period, the UP counter overflows  $M/(2K)$  times, and hence produces  $M/(2K)$  carries. The same number of borrows are generated by the DOWN counter as well. The  $M/(2K)$  carries cause the next positive edge of the  $u_2'$  signal to be advanced by  $M/(2K)$  ID clock cycles, where one ID clock cycle lasts for  $1/(2Nf_0)$  seconds. It follows that the duty cycle of  $u_2'$  can vary in the range

$$0.5\left(1 - \frac{M}{2KN}\right) < \delta < 0.5\left(1 + \frac{M}{2KN}\right) \quad (11.15)$$

When  $K$  is chosen smaller than  $M/2$ , the UP counter produces more than one carry in one UP-counting period, which of course increases the ripple on the output signal. To get minimum ripple, we should choose

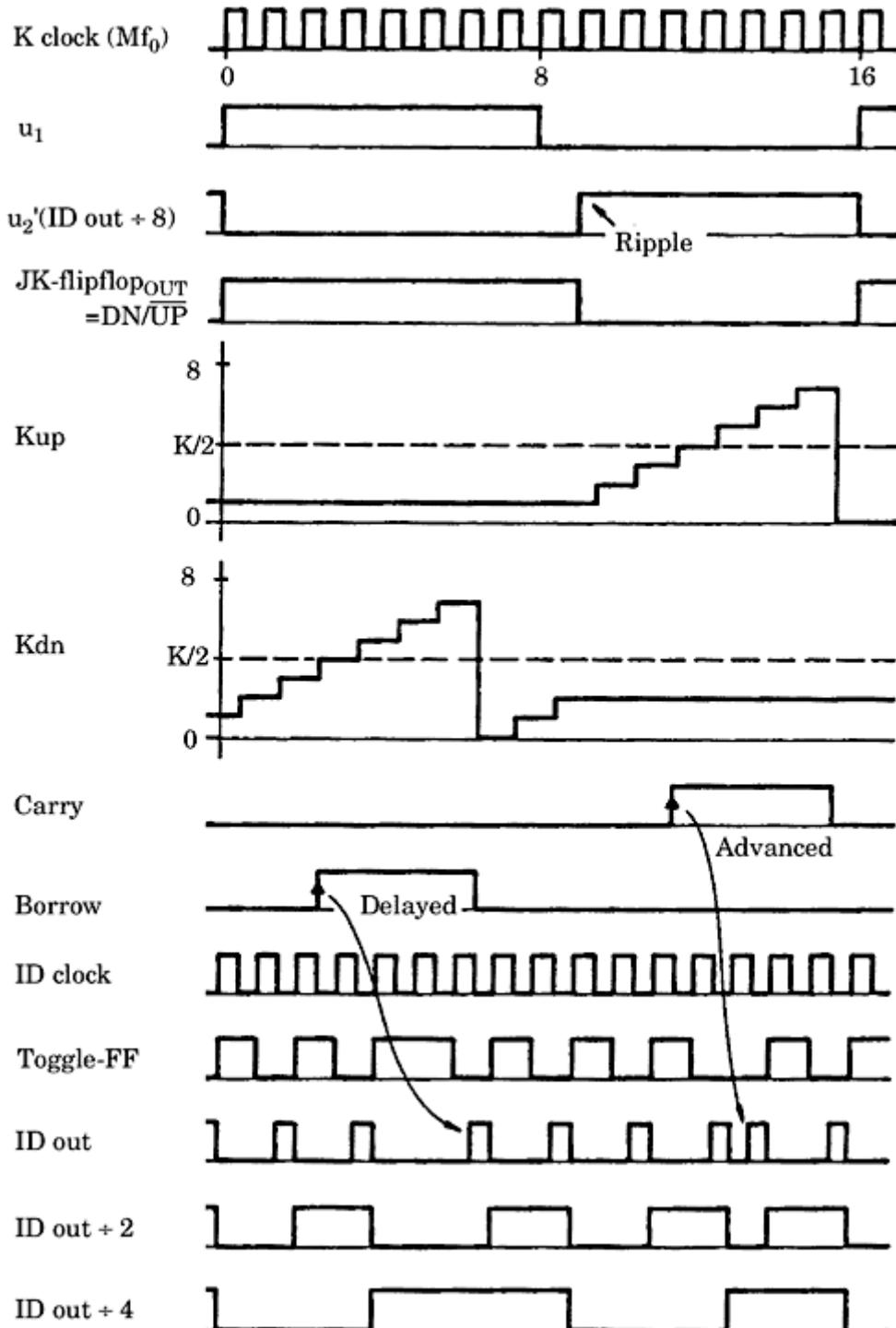
$$K = \frac{M}{2} \quad (11.16)$$

for the JK-flipflop PD. In contrast to the EXOR phase detector, the waveform of  $u_2'$  is unsymmetrical even if  $K$  is specified for minimum ripple. In many cases,  $M = 2N$ —in other words, the  $K$  clock and the ID clock are taken from the same generator. Then the duty cycle is in the range

$$(11.17)$$

$$0.5\left(1 - \frac{1}{K}\right) < \delta < 0.5\left(1 + \frac{1}{K}\right)$$

Selecting a large value for  $K$  reduces the ripple accordingly.



**Figure 11.17** Like Fig. 11.15, but the JK-flipflop phase detector is used. The ADPLL operates at its center frequency. The parameters of the circuit are  $M = 16$ ,  $K = 8$ , and  $N = 8$ .

### The hold range of the ADPLL

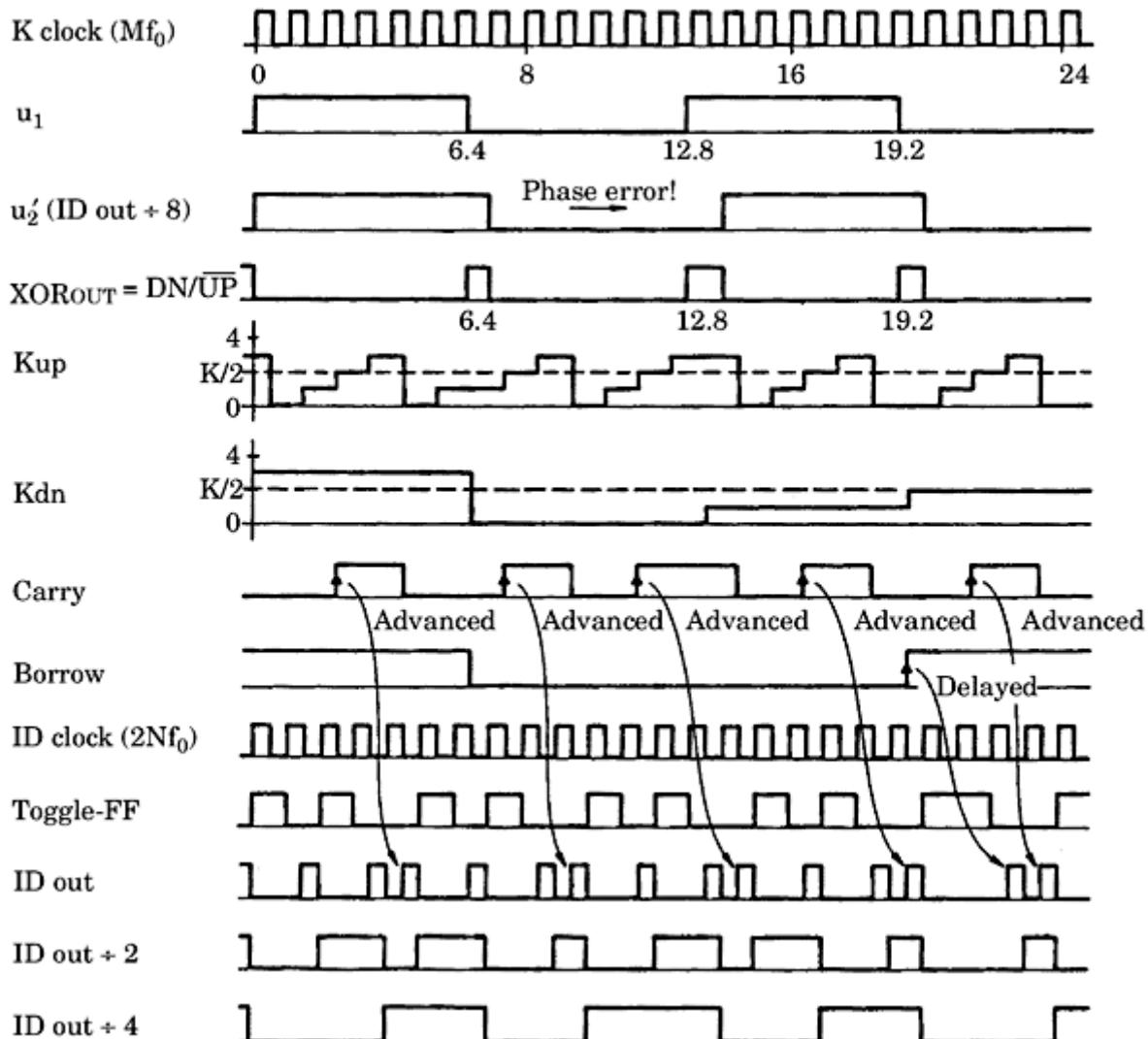
It is time now to see what happens if the frequency  $f_1$  of the reference signal deviates from the center frequency  $f_0$ . We assume first that an EXOR is used as the phase detector. Furthermore, let the reference frequency be  $f_1 = 1.25 f_0$ . [Figure 11.18](#) shows the waveforms for the case  $M = 16$ ,  $K = 4$ , and  $N = 8$ . For the ADPLL to generate a higher frequency, the UP counter must operate during longer time intervals than the DOWN counter. The phase error must therefore

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).

Copyright ©2004 The McGraw-Hill Companies. All rights reserved.

Any use is subject to the Terms of Use as given at the website.



**Figure 11.18** As in Fig. 11.15, the EXOR phase detector is used. The reference frequency is 1.25 times the center frequency here. Consequently, the UP-counting section of the  $K$  counter is active most of the time, and more carries than borrows are generated on average.

become positive. As Fig. 11.18 shows, the signals  $u_1$  and  $u_2'$  are nearly in phase now, which corresponds to a phase error near  $90^\circ$ . The UP counter is active most of the time, and many more carries than borrows are produced. This forces the ID counter to increase its output frequency.

A simple consideration yields the range of frequencies the ADPLL can work with. It is clear that the ADPLL generates the maximum output frequency when the  $K$  counter is continually counting up. The frequency of carry pulses then is given by

$$f_{\max} = f_0 \frac{M}{K} \quad (11.18)$$

Because each carry applied to the increment input of the ID counter causes 1/2 cycle to be added to the IDout signal (refer to [Sec. 11.1.3](#) and [Fig. 11.10](#)), the frequency at the output of the ID counter is increased by

$$\Delta f_{\text{IDout}} = f_0 \frac{M}{2K} \quad (11.19)$$

Because the  $\div N$  counter scales down that frequency by  $N$ , the maximum frequency deviation from the center frequency the ADPLL can handle is

$$\Delta f_H = f_0 \frac{M}{2KN} \quad (11.20)$$

This is nothing else than the *hold range* of the ADPLL. The hold range given by Eq. (11.20) is only realizable, however, if  $N$  is chosen larger than  $N_{\min}$  defined in Eq. (11.13). If  $N$  is smaller than  $N_{\min}$ , some of the carries and borrows are overslept, and the hold range is limited to

$$\Delta f_H = \frac{f_0}{3} \quad (11.21)$$

A simple consideration shows that the hold range given by Eq. (11.20) or (11.21) is a theoretical limit, which is not realized in practice. For the parameters  $M = 16$ ,  $K = 4$ ,  $N = 8$ , and for an EXOR phase detector, we obtain a theoretical hold range of  $\Delta f_H = 1.25 f_0$ . This is exactly the situation depicted in Fig. 11.18. We should note, however, that the duration of one cycle of the reference signal is now  $4/5$  of the duration of the reference period  $1/f_0$ , or in other words, the duration of one reference cycle is 12.6 cycles of the  $K$  clock. Assuming that the reference signal performs a positive edge on the 0th  $K$  clock, the next edges occur after 6.4, 12.8, 19.2, and so on,  $K$  clocks. Thus, the edges of  $u_1$  generally do not coincide with the edges of the output signal  $u_2'$ . As a consequence, the  $K$  counter is not continually counting up, but there are short intervals where the DOWN counter becomes active. Borrow pulses occur from time to time, as seen in Fig. 11.18. We see very clearly from the waveform of  $u_2'$  that the phase error increases from cycle to cycle. Hence, the ADPLL is not able to operate at the limit of the hold range. The author does not know an exact method to calculate the usable frequency range of an ADPLL, but computer simulations have shown that the maximum frequency deviation comes close to the hold range, say to about 90 percent of it.

From the LPPLS and DPLLS, it is well known, however, that the useful frequency range is not given by the hold range but rather by the lock-in or pull-in ranges. The question arises here whether such parameters can also be defined for the ADPLL under concern. Also here, the author cannot give an answer which is theoretically justified.

As the following analysis of phase-transfer function indicates, this type of ADPLL is a first-order loop whose time constant is on the order of the period of the reference signal—hence, a very fast system. Simulations show that even for very large frequency steps applied to the reference input, the ADPLL does not lock out, so for the practitioner it seems affordable to state that lock-in range, pull-out range, pull-in range, and hold range are all about the same for this circuit. We perform some ADPLL simulations in Chap. 12.

It can be observed that the output signal  $u_2'$  exhibits ripple or phase jitter whenever the reference frequency  $f_1$  deviates from the center frequency  $f_0$ . If the quotient  $f_1/f_0$  is a rational fraction

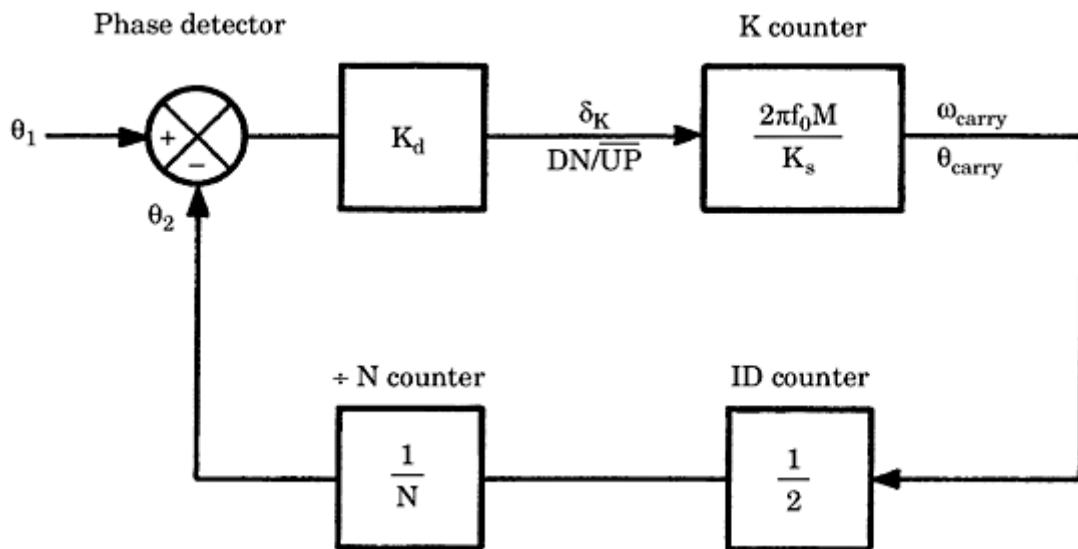
$$\frac{f_1}{f_0} = \frac{m}{n}$$

where  $m, n = \text{integer}$ , and then we have  $mT_0 = nT_1$ , where  $T_0 = 1/f_0$  and  $T_1 = 1/f_1$ —thus,  $n$  cycles of the reference signal have exactly the same duration as  $m$  cycles of a signal whose frequency equals the center frequency. In this case, the ripple pattern of the  $u_2'$  signal becomes periodic. If  $f_1/f_0$  is not rational, however, the ripple pattern does not repeat itself.

### Frequency-domain analysis of the ADPLL

As with the LPLL and the DPLL, it is possible to derive the phase transfer function and the error-transfer function for the ADPLL. To get the phase transfer function, a mathematical model of the ADPLL (Fig. 11.13) must be found. The model is shown in Fig. 11.19. The phase detector represents a zero-order block with gain  $K_d$ . The phase detector output signal controls the duty factor  $\delta_K$  of the  $K$  counter. This duty factor is defined by the average fraction of time the UP counter is active. Thus, for  $\delta_K = 1$ , the UP counter is permanently active, whereas for  $\delta_K = -1$ , the DOWN counter is permanently active. If the EXOR phase detector is used,  $\delta_K$  will be 1 for a phase error of  $\theta_e = \pi/2$  and -1 for  $\theta_e = -\pi/2$ . For the EXOR, we then have

$$K_d = \frac{2}{\pi} \quad (11.22a)$$



**Figure 11.19** Mathematical model of the ADPLL. Definitions of the symbols are in the text.

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

For the JK-flipflop, the phase detector gain is

$$K_d = \frac{1}{\pi} \quad (11.22b)$$

In a number of other texts, the phase error is alternatively specified in cycles (of the reference period  $1/f_1$ ) and not in radians. Here, the phase detector gains become 4 for the EXOR and 2 for the JK-flipflop.

We prefer, however, to specify all phase signals in radians in order to get the required phase-transfer function. Now the mathematical model of the  $K$  counter must be found. As explained in [Sec. 11.3.2](#), the number of carry pulses generated per second is given by

$$f_{\text{carry}} = \delta_K \frac{Mf_0}{K} \quad (11.23)$$

The corresponding angular frequency  $\omega_{\text{carry}}$  is therefore

$$\omega_{\text{carry}} = \delta_K 2\pi \frac{Mf_0}{K} \quad (11.24)$$

Because we are looking for the phase transfer function, we must know the phase  $\theta_{\text{carry}}$  of the  $K$  counter output signal. Because the phase is simply the integral of angular frequency over time, the phase transfer function of the  $K$  counter becomes

$$K_K(s) = \frac{\Theta_{\text{carry}}(s)}{\Delta_K(s)} = \frac{2\pi Mf_0}{Ks} \quad (11.25)$$

where  $\Delta_K(s)$  and  $\Theta_{\text{carry}}(s)$  are the Laplace transforms of the signals  $\delta_K$  and  $\theta_{\text{carry}}$ , respectively. Because each carry pulse applied to the increment input of the ID counter causes 1/2 cycle to be added to the IDout signal, the ID counter can be modeled simply by a zero-order block having the gain 1/2. Clearly, the  $\div N$  counter is a block with gain 1/ $N$ . Having the model, the phase transfer function  $H(s)$  is now found to be

$$H(s) = \frac{\omega_0}{s + \omega_0} \quad (11.26)$$

where  $\omega_0$  is given by

$$\omega_0 = \frac{K_d \pi M f_0}{K N} \quad (11.27)$$

Moreover, the error-transfer function  $H_e(s)$  is

$$H_e(s) = \frac{s}{s + \omega_0} \quad (11.28)$$

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

Clearly, this ADPLL is a first-order system. Its time constant is given by

$$\tau = \frac{1}{\omega_0} = \frac{KN}{K_d \pi M f_0} \quad (11.29)$$

Thus, for the EXOR phase detector, the time constant of the ADPLL becomes

$$\tau(\text{EXOR}) = \frac{KN}{2Mf_0} \quad (11.30a)$$

and for the JK-flipflop phase detector

$$\tau(JK) = \frac{KN}{Mf_0} \quad (11.30b)$$

If the  $K$  modulus is chosen for minimum ripple—that is,  $K = M/4$  for the EXOR and  $K = M/2$  for the JK-flipflop PD—we obtain  $\tau = (N/8)T_0$  for the EXOR and  $\tau = (N/2)T_0$  for the JK-flipflop PD, where  $T_0 = 1/f_0$ . This shows that for small divider ratio  $N$ , the ADPLL settles extremely fast. Only for large  $N$  does its response to phase or frequency steps become slower. Some numerical examples will be presented in [Chap.12](#).

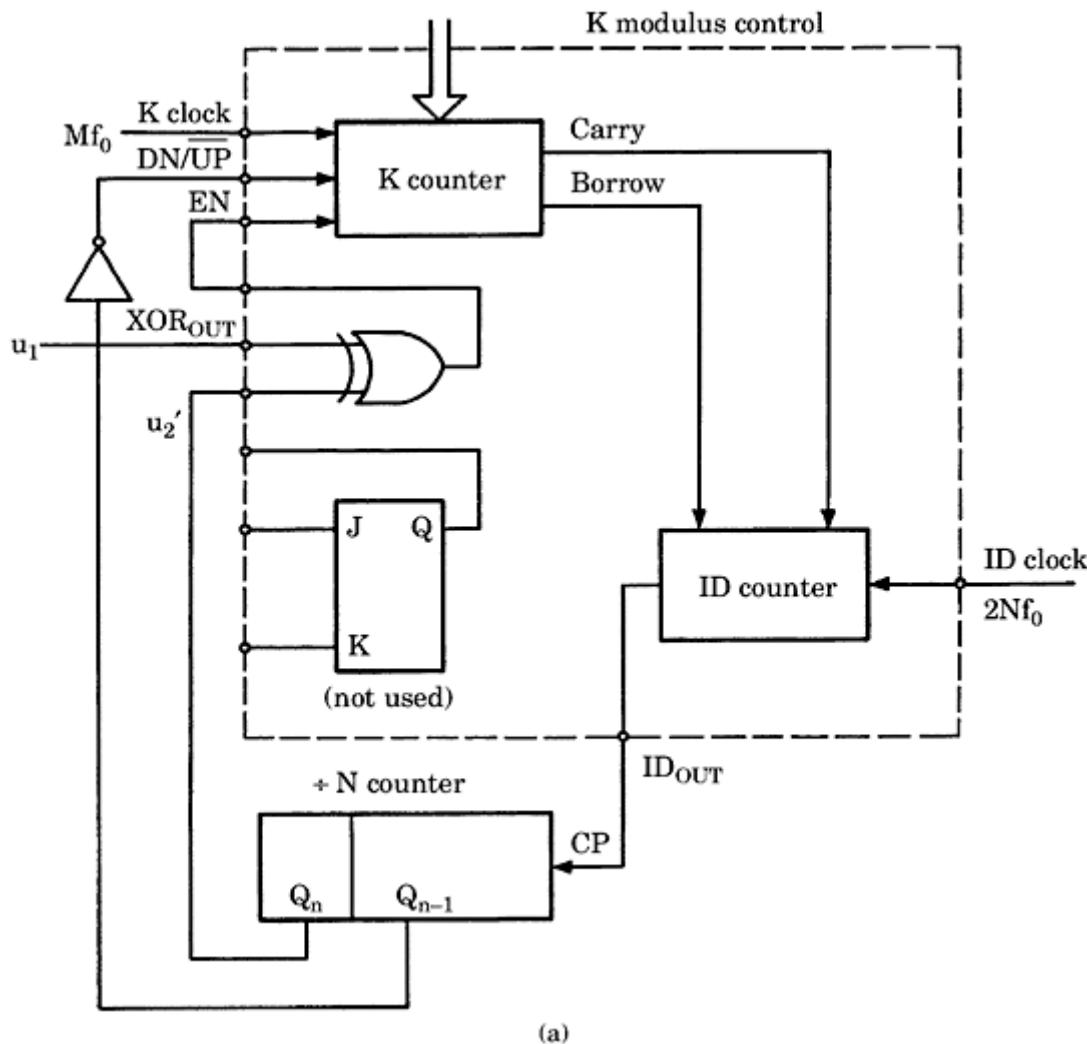
## Ripple reduction techniques

We saw in [Sec. 11.3.1](#) that ripple in ADPLLs can be minimized by choosing an optimum value for the  $K$  counter modulus—in other words,  $K = M/4$  when the EXOR PD is used or  $K = M/2$  when the JK-flipflop PD is used. Other simple ways are available to reduce ripple. [Figure 11.20a](#) shows a ripple cancellation scheme that uses a feature of the  $K$  counter that has not yet been mentioned: the ENABLE input of the  $K$  counter. Only one additional inverter is required in this circuit. The  $K$  counter is in operation only when the ENABLE input  $EN$  is high. To suppress ripple, the second most significant bit of the  $\div N$  counter is used in addition (denoted  $Q_{n-1}$ ). The frequency at the  $Q_{n-1}$  output is twice the frequency of the output signal  $u_2'$ . In this circuit, the EXOR phase detector is utilized. Its output is not connected, however, to the  $DN/\overline{UP}$  input of the  $K$  counter, but rather to its ENABLE input  $EN$ . The  $DN/\overline{UP}$  input is driven by the  $Q_{n-1}$  signal now. This forces the signals  $u_1$  and  $u_2'$  to be nearly in phase when the ADPLL operates at its center frequency ([Fig. 11.20b](#)). If this is the case, the EN input is FALSE most of the time, so neither the UP counter nor the DOWN counter is active.

Only when the reference frequency deviates from the center frequency is a phase difference between  $u_1$  and  $u_2-$  is established.

[Figure 11.20c](#) shows the case for maximum phase error  $\theta_e$ . Now, the  $EN$  signal is high during about 50 percent of the time. At the same time, the UP counter becomes active, so the  $K$  counter generates the maximum number of carries. As we easily recognize from the waveforms, the average number of carries produced is only about half that number for a normal ADPLL circuit ([Fig. 11.13](#)).

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.



**Figure 11.20** Ripple cancellation scheme for an ADPLL. (a) This circuit makes use of the ENABLE input of the  $K$  counter. For details, refer to the text. (b) The ADPLL operates on its center frequency. (c) The reference frequency is higher than the center frequency.

Consequently, the hold range is reduced roughly by a factor of 2. It has been shown<sup>16</sup> that the hold range of the circuit becomes

$$\Delta f_H = \frac{Mf_0}{2N(2K + 1)} \quad (11.31)$$

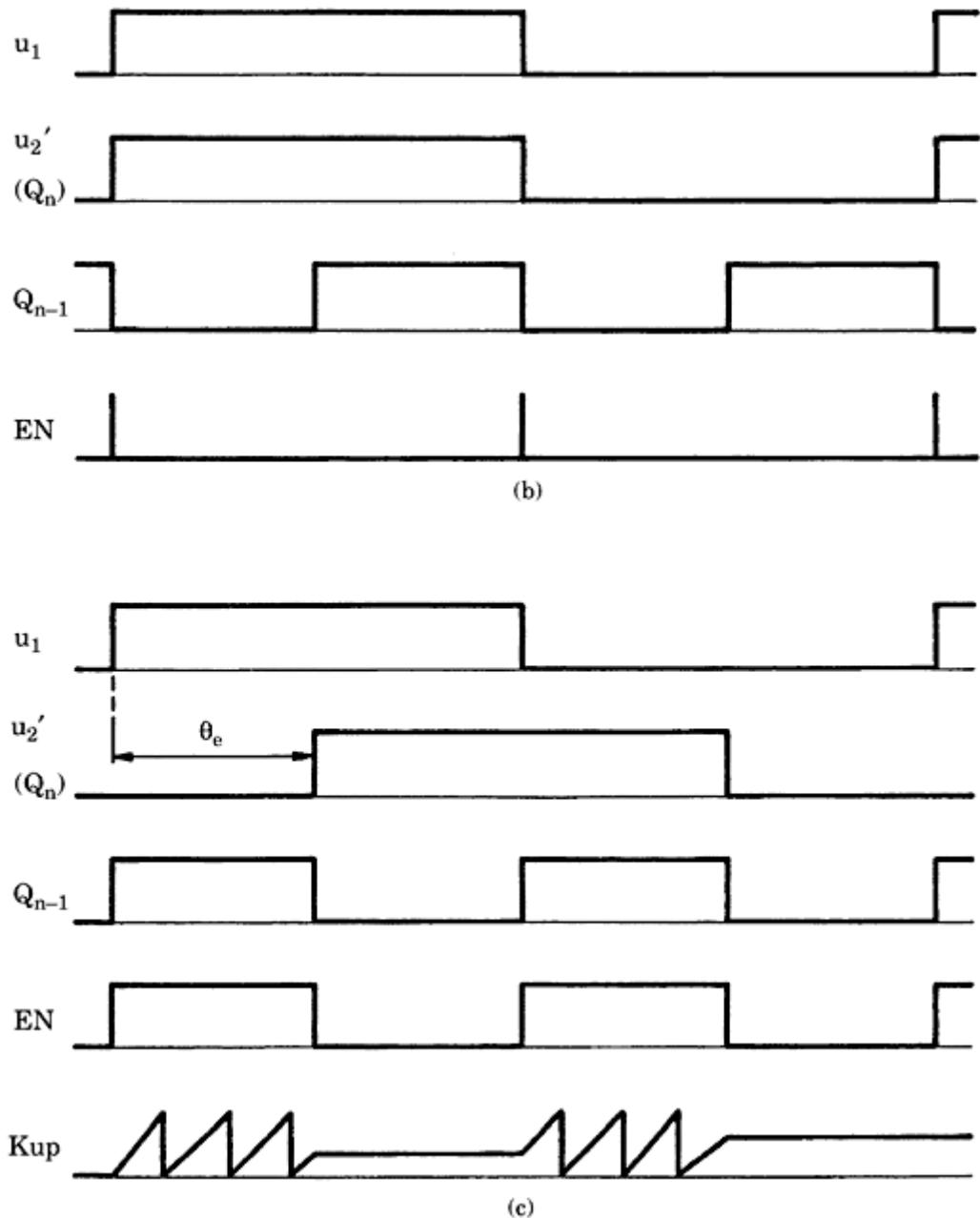
If  $M = 2N$  and  $K$  is large, this reduces to

$$\Delta f_H \approx \frac{f_0}{2K} \quad (11.32)$$

Still other ripple cancellation schemes are discussed in .

## Higher-order ADPLLs

The ADPLL considered in this section is a first-order system. As we have seen in Eqs. (11.30a) and (11.30b), its settling time can be made extremely short. This can be an advantage in some applications, but there are other cases where a slower



**Figure 11.20 (Continued)**

response—combined with better noise-suppression capability—is desired. If two ADPLLs are cascaded, a second-order ADPLL is obtained. This arrangement has been considered by Rosink<sup>16</sup> in more detail.

### Typical ADPLL Applications

Because of the availability of low-cost ADPLL ICs, this type of PLL can replace the classical DPLL in many applications today. The ADPLL is mainly used in the field of digital

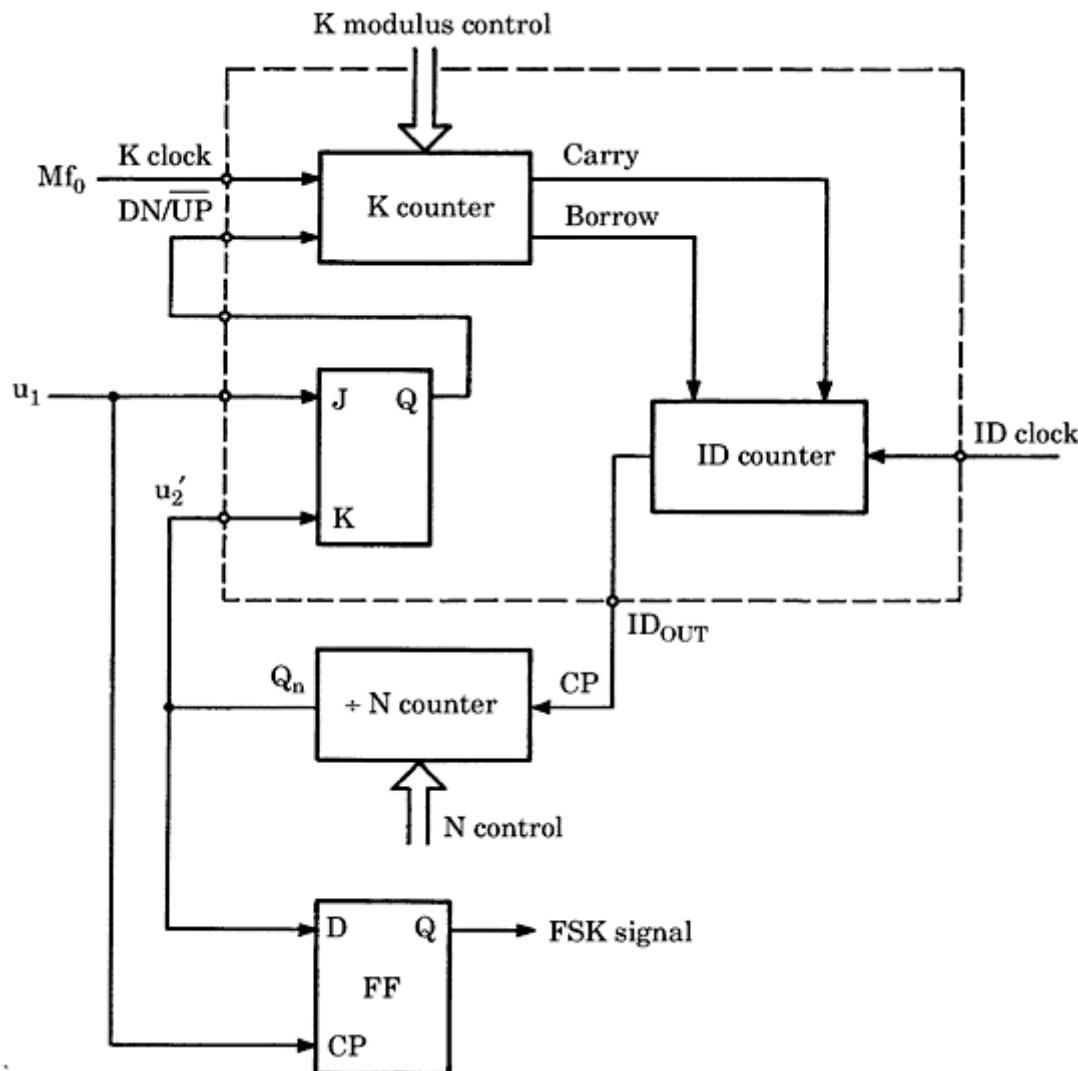
communications. A good example is the FSK decoder, which we shall now explore. FSK stands for “frequency shift keying.” In FSK data transmission, serial binary data are transmitted using two different frequencies, one of which represents a logical 0, and the other a logical 1.

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

An extremely simple FSK decoder circuit is shown in Fig. 11.21. The center frequency  $f_0$  of the ADPLL is chosen such that it is between the two frequencies used by the FSK transmitter. Because the JK-flipflop phase detector is utilized here, the two signals  $u_1$  and  $u_2'$  would be exactly in antiphase if the ADPLL were operating at its center frequency (see also Fig. 11.17). Therefore, the contents of the divided-by- $N$  counter would be just  $N/2$  at the time where the reference signal  $u_1$  performs a positive transition. If the reference frequency is greater than the center frequency, the output signal  $u_2'$  must lead the reference signal  $u_1$  in order to enable the  $K$  counter to produce more carries than borrows on the average. Consequently, the contents of the divide-by- $N$  counter will be greater than  $N/2$  at the positive transient of  $u_1$ .

If the reference frequency is lower than the center frequency, however, the reverse is true, and the contents of the divide-by- $N$  counter is less than  $N/2$  at that instant of time. Because the output signal  $u_2'$  is low for a content less than  $N/2$ , and high for a content equal to or more than  $N/2$ , the information transmitted by the FSK signal can be recovered simply by storing the  $u_2'$  signal in a



**Figure 11.21** The FSK decoder using the 74HC/HCT297 ADPLL IC. An additional D-flipflop is used to deliver the demodulated FSK signal.

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

D-flipflop every time the reference signal goes high. This D-flipflop is shown at the bottom of Fig. 11.21. In Sec. 11.5, we discuss a design procedure for ADPLLs and use it to specify the parameters of this application. Many other ADPLL applications are discussed on the data sheets of the 74HC/ HCT297<sup>31</sup> and in various application notes delivered by the IC suppliers.<sup>9,16</sup>

## Designing an ADPLL

Designing an ADPLL is much easier than designing an LPLL or DPLL, because we have to specify only the three parameters  $M$ ,  $K$ , and  $N$ . For the 74HC/HCT297,  $K$  must always be an integer power of 2 and can be selected in the range  $2^3$  to  $2^{17}$ . For the divide-by- $N$  counter (refer to Fig. 11.13, for example), a straight-binary counter is used in most cases, so  $N$  will usually be an integer power of 2, too. Moreover, to use the same signal generator for the  $K$  clock and for the ID clock, we set  $M = 2N$  whenever possible. Consequently, all ADPLL parameters are integer powers of 2 in most cases.

The selection of the phase detector represents a further degree of freedom. Because the JK-flipflop phase detector is edge-sensitive, it can only be used in applications where no cycles of the reference signal are missing. Otherwise, the output of the JK-flipflop would hang up in the 1 or 0 state all the time where the reference signal disappears, which inevitably would lock out the loop.

As we have seen in Sec. 11.3, there are a number of interdependences among the quantities  $M$ ,  $K$ , and  $N$ . If minimum ripple is a goal,  $M$  should be chosen to be  $4K$  when the EXOR PD is used, or  $2K$  when the JK-flipflop is used. Given the ratio  $M/K$ , the only parameter that can be freely selected is  $N$ . To avoid “over-sleeping” of carries and borrows,  $N$  should be greater than a minimum value  $N_{\min}$ , which is given by Eq. (11.13). As indicated by Eq. (11.20), the resulting hold range is a function of  $M$ ,  $K$ , and  $N$ . Furthermore, the settling time  $\tau$  of the ADPLL also depends on these parameters, as seen from Eqs. (11.30a) and b).

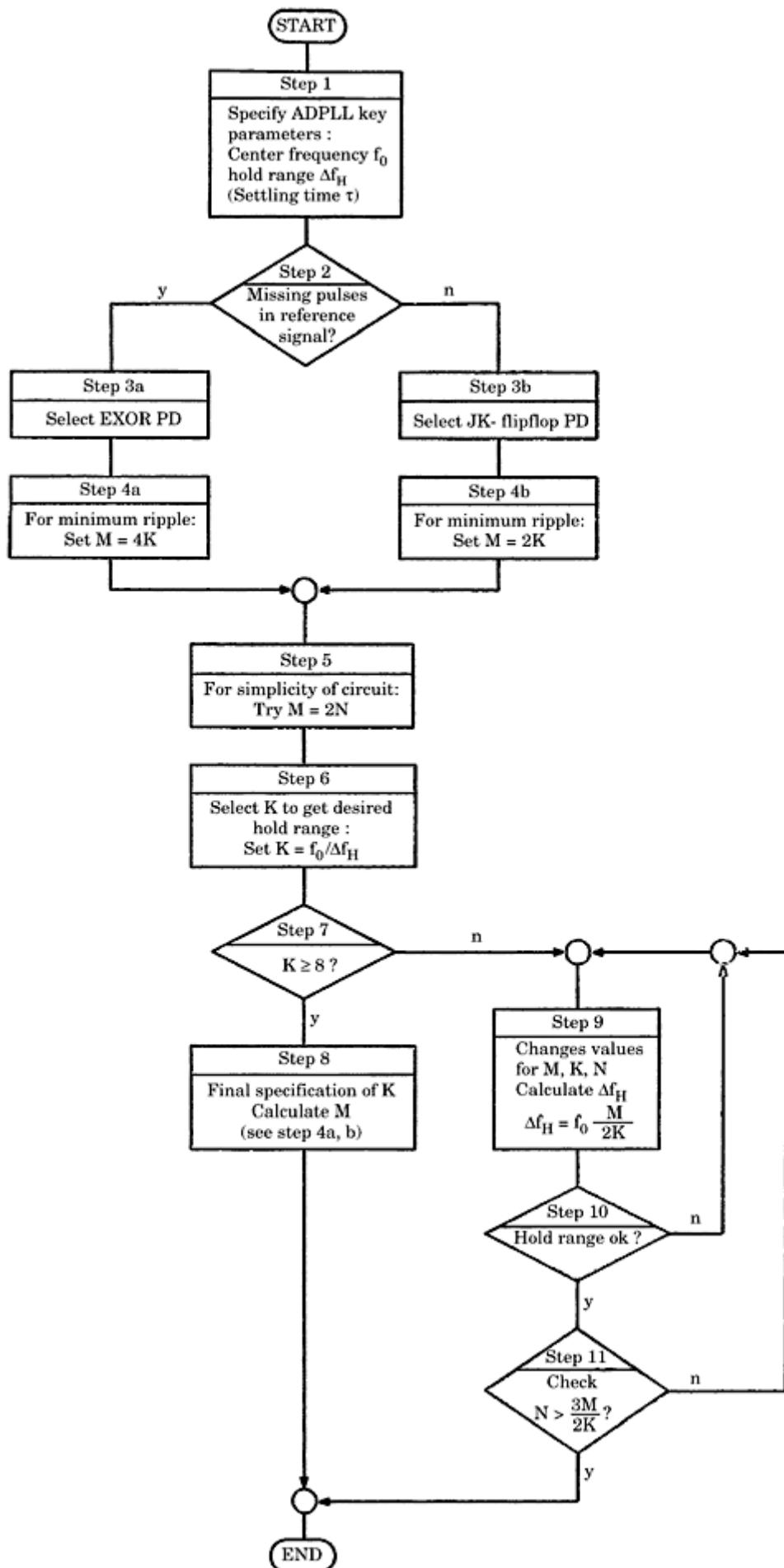
On the basis of these general considerations, we can derive the design procedure given in Fig. 11.22. Like the procedures worked out for the LPLL and the DPLL, this program should not be considered as a universal recipe for every kind of ADPLL but rather as a checklist. To demonstrate the design procedure, we now implement the FSK decoder described in Sec. 11.4 (refer also to Fig. 11.21).

### Case study: designing an ADPLL FSK decoder

**Step 1.** To specify the parameters of this ADPLL system, we must first define its center frequency and hold range. Let’s assume that the FSK transmitter uses the frequencies  $f_{11} = 2100$  Hz and  $f_{12} = 2700$  Hz to encode the binary information 0 and 1, respectively. Following the description in Sec. 11.4, we choose a center frequency of  $f_0 = 2400$  Hz. To ensure that both frequencies of the FSK transmitter are within the hold range of the ADPLL, we specify a hold range of  $\Delta f_H = 600$  Hz. We do not specify a settling time  $\tau$  for the moment but will check at the end of the design whether the resulting settling time is appropriate or not.

**Any use is subject to the Terms of Use as given at the website.**





**Figure 11.22** The design procedure for the ADPLL.

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

**Step 2.** The decision of whether the EXOR or the JK-flipflop will be used as the phase detector has been made in advance. Keep in mind, however, that this ADPLL relies on an input signal that does not reveal missing pulses.

**Step 3b.** The JK-flipflop is used as phase detector.

**Step 4b.** For minimum ripple, we choose  $M = 2K$ .

**Step 5.** To get the simplest circuit, we choose  $M = 2N$ . The clocks for the  $K$  counter and for the ID counter are identical then and can be taken from the same signal source.

**Step 6.** This step reveals the first flop in our design, because  $K$  should be chosen as 4 to get the desired hold range of 600 Hz. The 74HC/ HCT297 circuit does not allow  $K = 4$ , but only  $K = 8, 16$ , and so on. The design proceeds with step 9.

**Step 9.** To get a hold range of 600 Hz, the assumption  $M = 2N$  cannot be made. If we keep  $M = 2K$  (the condition for minimum ripple), the hold range can be expressed as

$$\Delta f_H = f_0 \frac{M}{2KN} = f_0 \frac{1}{N} \quad (11.33)$$

The hold range thus becomes 600 Hz if we specify  $N = 4$ . Choosing the minimum value for  $K$  ( $K = 8$ ), we get  $M = 16$ . Hence, we must insert an additional JK-flipflop between the  $K$  clock and the ID clock that scales down the  $K$  clock frequency by a factor of 2.

**Step 10.** The hold range is okay now (600 Hz).

**Step 11.** To avoid overslept carries and borrows,  $N$  must be larger than the minimum value  $N_{\min} = 3M/2K$ . Because  $N_{\min} = 3$  in our case,  $N = 4$  is a valid choice.

The design is completed now. Finally, we check the settling time  $\tau$  of this circuit. Using Eq. (11.30b), we get  $\tau = 2/f_0 = 2T_0$ , where  $T_0$  is the duration of one reference cycle. This indicates that the ADPLL settles within two cycles of the reference signal, which is certainly acceptable in this application. When simulating the ADPLL on the PC (cf. Chap. 12), we take a look at this circuit and see how it performs.

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

# Computer-Aided Design and Simulation of ADPLLs

## Designing the ADPLL

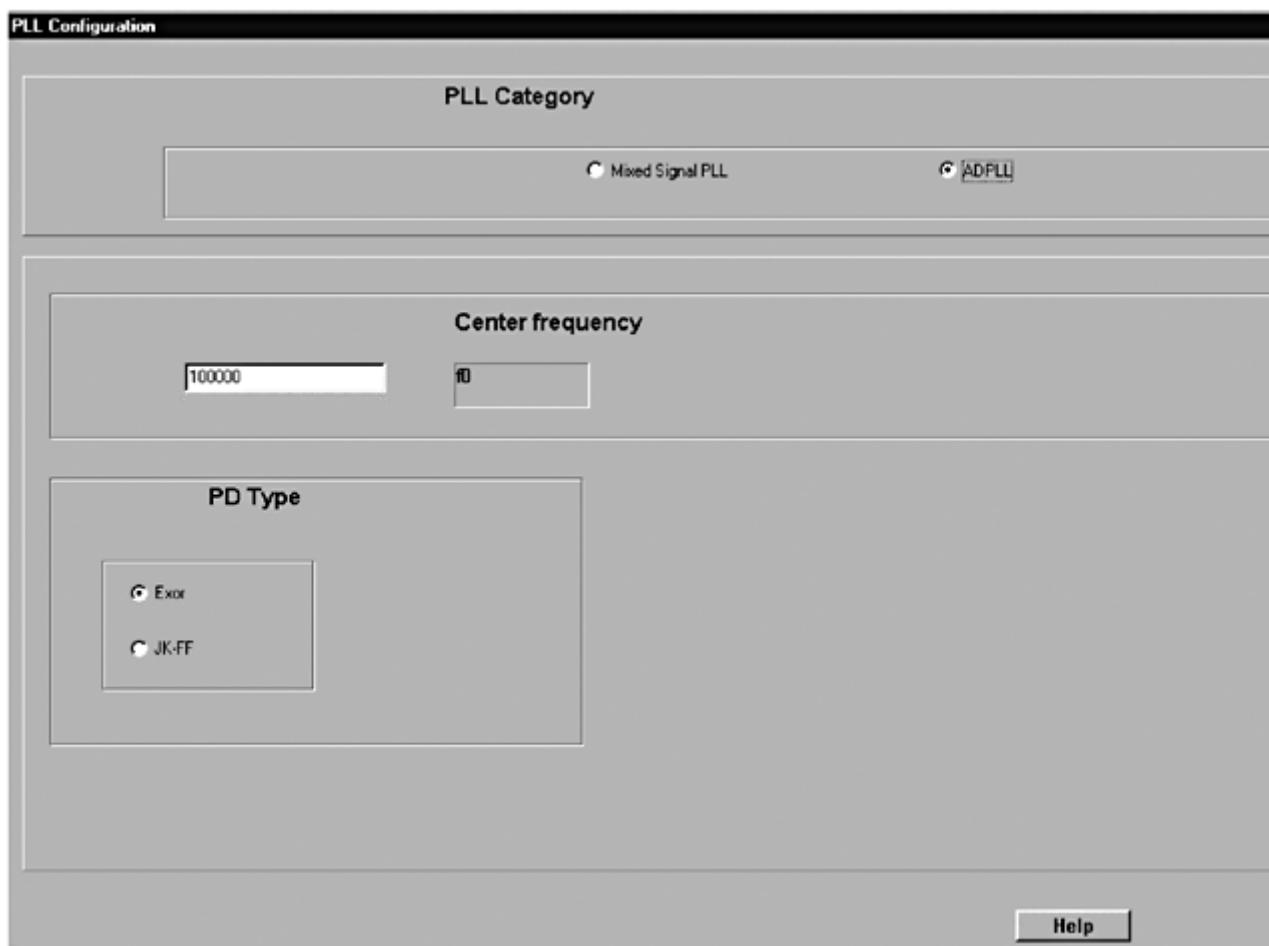
The program distributed with this book can also be used to design and simulate ADPLLs. The procedures are similar to those described in [Chap. 10](#) and are best explained by a quick tour.

Start the program and click the CONFIG speed button in the taskbar. This brings up the configuration dialog box, shown in [Fig. 12.1](#). In the panel PLL Category, click the ADPLL radio button. To configure an ADPLL, it is only required to specify the center frequency  $f_0$  and the type of phase detector: EXOR or JK-flipflop. Click the Done button to complete the dialog.

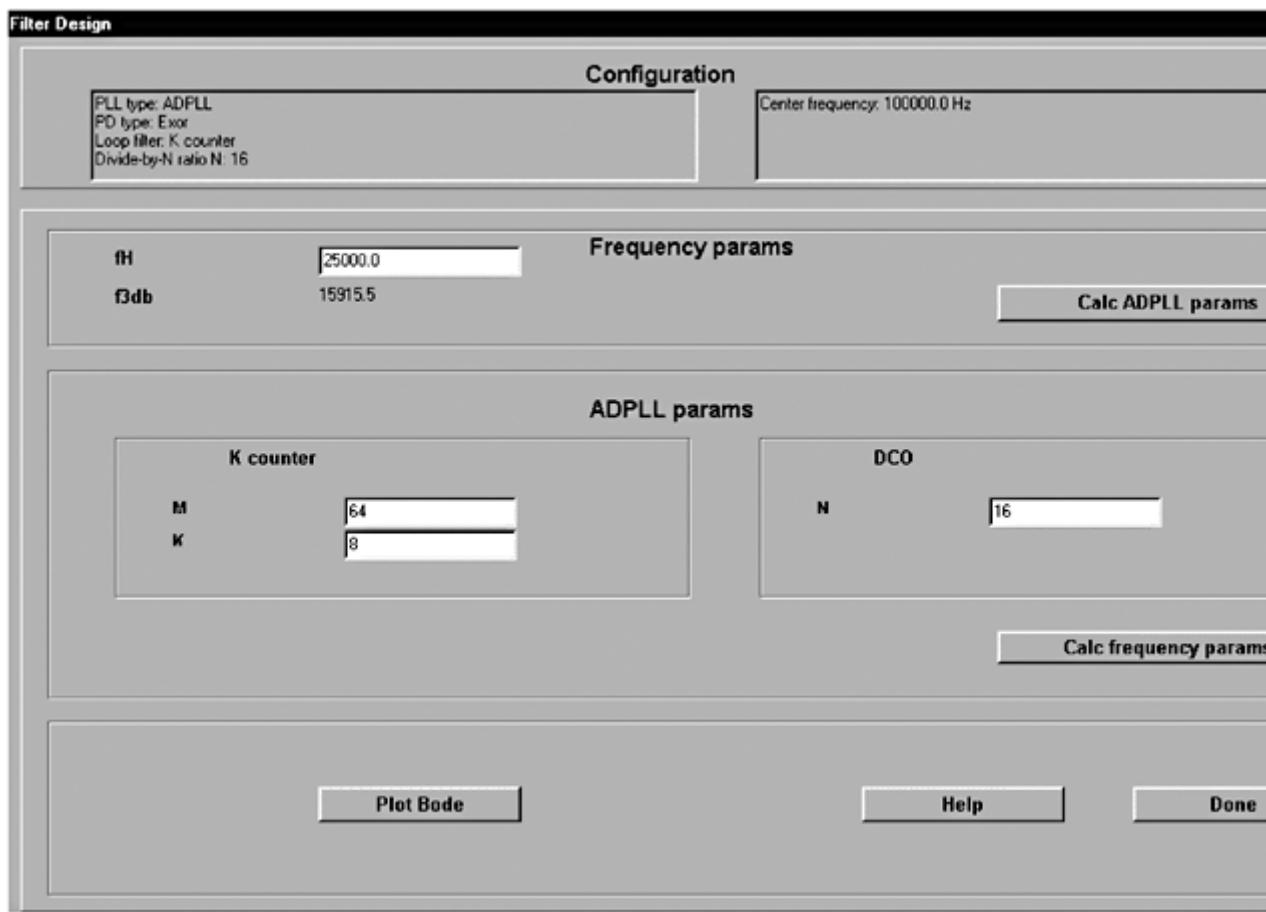
Next, click the DESIGN speed button in the taskbar. This activates the *Filter Design* dialog box, [Fig. 12.2](#). In the topmost panel (Configuration), the actual ADPLL configuration is displayed. Below it are two panels labeled Frequency Params and ADPLL Params, respectively. The ADPLL params are represented by the parameters  $K$ ,  $M$ , and  $N$  as described in [Sec. 11.3](#). Based on the last specified values for  $K$ ,  $M$ , and  $N$ , the program computes the corresponding ADPLL hold range  $\Delta f_H$  and the 3-dB corner frequency  $f_{3dB}$ . As in the case of the mixed signal PLL, the user can proceed in two ways.

He/she enters the desired hold range into the edit window labeled  $fH$  and then clicks the Calc ADPLL Params button. The program then computes the parameters  $K$ ,  $M$ , and  $N$  required to meet that goal. Or, alternatively, the user can immediately enter values for  $K$ ,  $M$ , and  $N$  into the corresponding edit windows and click the Calc Frequency Params button. The program then computes the resulting hold range ( $fH$ ) and 3-dB corner frequency ( $f_{3dB}$ ).

When the user chooses the first of these options and enters the desired hold range, the program calculates the appropriate values for  $K$ ,  $M$ , and  $N$ , using [Eq. \(11.20\)](#). Because we have only one equation for three unknown parameters, two of them can be chosen arbitrarily. The program therefore sets  $K = 8$  and  $N = 16$ .



**Figure 12.1** The dialog box for configuring an ADPLL.



**Figure 12.2** The dialog box for ADPLL parameter specification.

---

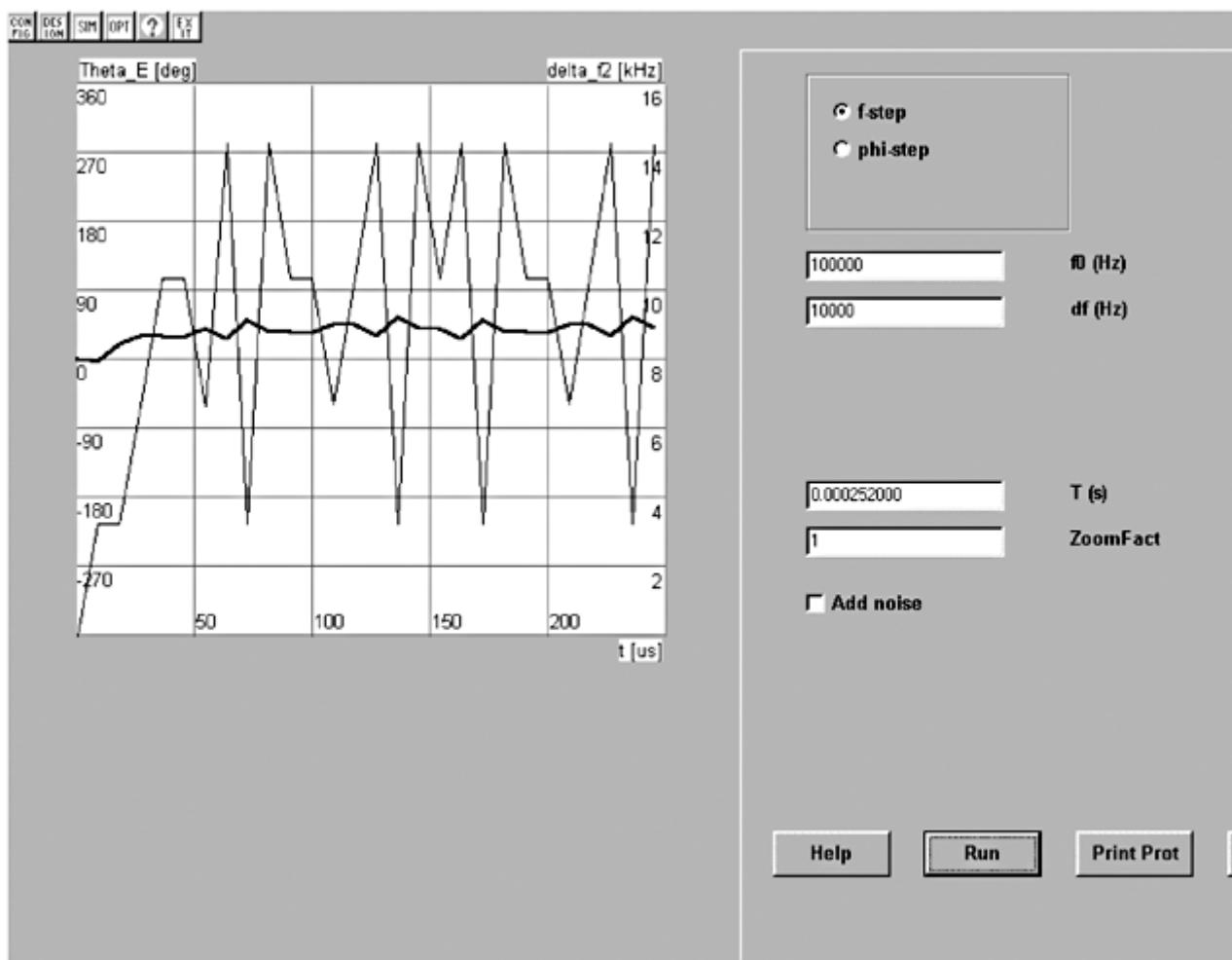
Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

and finally computes the appropriate value of  $M$ . This is not necessarily an optimum choice. Eventually, the ripple of the ADPLL could be too large. To reduce the ripple, the user would probably increase  $N$ —for example, by setting  $N = 32$ . Because the hold range is proportional to  $M/(KN)$ ,  $M$  must be increased by the same factor—that is, by the factor 2.

The DESIGN dialog box also includes an option to display the Bode diagram of the open-loop and closed-loop gain. The Bode diagram is obtained by clicking the Bode button. The DESIGN dialog exits when the Done button is clicked. Thus, we are now ready to perform simulations.

## Simulating ADPLL Performance

To start ADPLL simulation, click the SIM speed button. This opens the simulation dialog box (see Fig. 12.3). The parameters of the simulation are entered in the panel on the right side. They are almost identical to those of Mixed Signal PLL simulations with the exception that there is no filter function here. In contrast to the mixed-signal PLL, there are no voltage signals to calculate. Instead of the signals  $u_d$  and  $u_f$ , the program computes in every reference cycle the instantaneous frequency  $f_2'$  of the output signal  $u_2'$  (cf. Fig. 11.13) and the



**Figure 12.3** Simulation of ADPLL performance.

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

instantaneous phase error  $\theta_e$ . The frame window on the left of the simulation dialog box displays  $\theta_e$  (thicker curve) and  $\Delta f_2$  (thinner curve), where  $\Delta f_2$  is the variation of output frequency—that is,  $\Delta f_2 = f_2' - f_0'$ , and  $f_0'$  is the scaled-down center frequency of the ADPLL. Due to the fact that the output signal  $u_2'$  can change state only on the edges of the clock signal (the ID clock in Fig. 11.13), only a discrete set of output frequencies can be created. As we see from the plot in Fig. 12.3, the output frequency varies in steps of approximately 3.4 kHz. Because one single value for  $\theta_e$  and  $\Delta f_2$  is calculated in each reference period, no filtering of these variables is performed.

As in the case of the mixed-signal PLL, noise can be added to the reference signal. This is done by checking the Add Noise checkbox and entering appropriate values for SNR(dB) and Noise BW(rel), as explained in Sec. 10.4 (cf. also Fig. 10.8).

In the next section, we will perform a number of case studies that will reveal the substantial differences in the behavior of ADPLLs when compared with the classical LPPLLs and DPLLs.

## Case Studies on ADPLL Behavior

In this section, we will investigate ADPLL performance using some typical applications.

**Case study 1.** *Dynamic performance of the ADPLL using the EXOR PD.* Start the simulation program, and click the CONFIG speed button. In the PLL Category panel, check the ADPLL radio button. In the Center Frequency panel, enter **100,000** into the  $f_0$  window. In the PD Type panel, check the EXOR radio button. Close the CONFIG dialog by clicking the Done button.

Click the DESIGN speed button. This opens the design dialog box. In the ADPLL Params panel, enter the values

$$\begin{aligned} K &= 8 \\ M &= 32 \\ N &= 16 \end{aligned}$$

Click the Calc Frequency Params button. The program computes the resulting hold range and 3-dB corner frequency. We get  $f_H = 12.5$  kHz and  $f_{3\text{dB}} = 7957$  Hz. Click the Done button to close the design dialog.

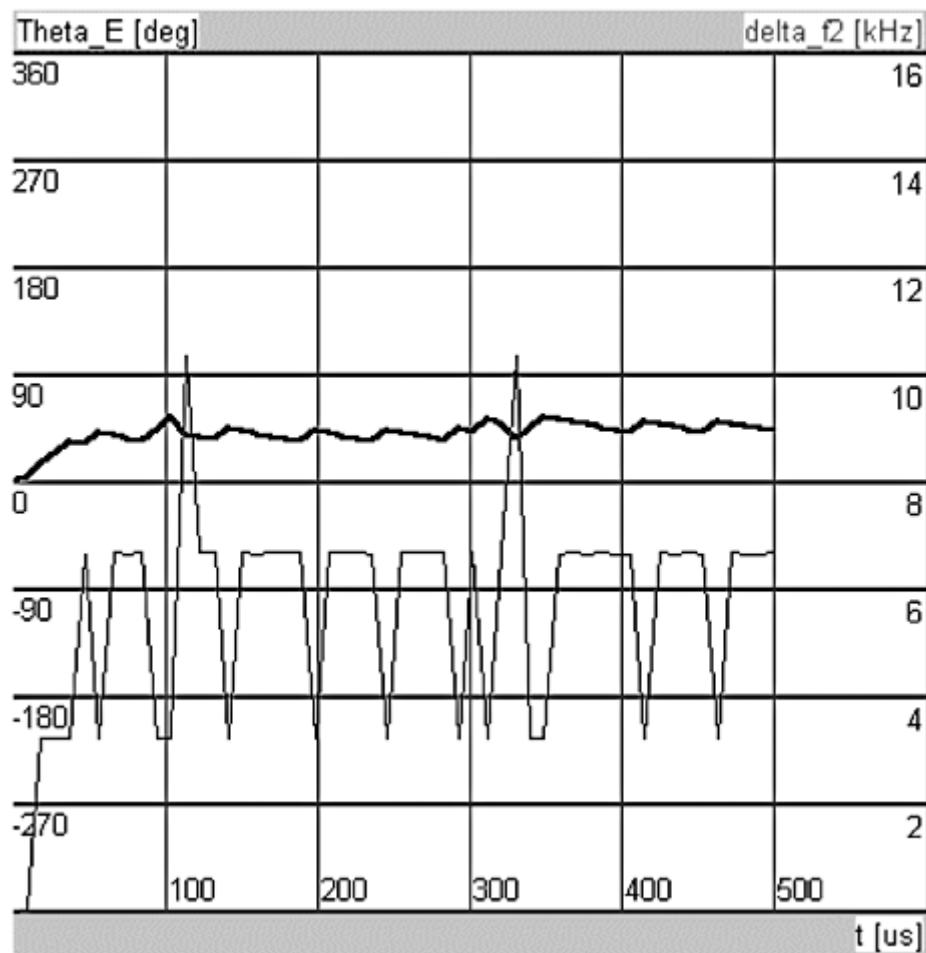
You are now ready for the simulation. Click the SIM speed button. Because we want to apply frequency steps to the reference inputs, check the f-Step radio button (cf. Fig. 12.3). In the edit window, enter the following parameters

$$\begin{aligned} f_0 &= 100,000 \text{ Hz} \\ df &= 6000 \text{ Hz} \\ T &= 0.000500 \text{ s} \\ \text{Zoom factor} &= 1 \\ \text{Add noise:} &\text{ unchecked} \end{aligned}$$

Select the RUN button. According to the theory surrounding this type of ADPLL, the time constant of the loop is  $\tau = 2 \cdot T_0 = 20 \mu\text{s}$ , where  $T_0 = 1/f_0$ . The frequency step chosen in this simulation is about half the lock range. The result of the simulation is shown in Fig. 12.4.

A linear system would settle to within 5 percent of its final state in about three time constants—in other words, in  $60 \mu\text{s}$ . The curve for the phase error is quite jittery, but the settling time is seen to be well in that range. The difference frequency curve looks very noisy indeed. Unlike a classical DPLL, the frequency of the output signal does not asymptotically approach the reference frequency but oscillates around that value. This is the most typical property of the ADPLL. By the nature of the ID counter, its output signal can only perform state changes on the edges of the ID clock signal. Hence, the instantaneous frequency  $f_2$  of the output signal can take only a number of discrete values. It is easily shown that  $f_2$  can take values of

$$f_2 = f_0 \left( \frac{2N}{2N \pm 1}, \frac{2N}{2N \pm 2}, \dots \right) \quad (12.1)$$



**Figure 12.4** An ADPLL simulation. Response to frequency step  $df = 6 \text{ kHz}$ . Thicker line = phase error; thinner line = frequency deviation.

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

For positive deviations,  $f_2$  can have the values 100, 103.2, 106.7... kHz. The curve for  $\Delta f_2$  indicates that the output frequency equals the reference frequency *on average*, however. Because the phase error curve is relatively smooth at about  $45^\circ$ , the ADPLL is firmly locked.

Let us increase now the frequency step to, say, 12 kHz. This simulation is shown in Fig. 12.5. The instantaneous output frequency now oscillates between 110 and 114 kHz approximately, and from the relatively quiet phase error curve we see that the system is still locked. The phase error is near its limit of stability—that is, slightly less than  $90^\circ$  on average. If the frequency step is made larger (13 kHz), the system can no longer maintain lock (see Fig. 12.6). The loop pulls the output frequency to about 114 kHz after some reference cycles, but because the phase error exceeds  $180^\circ$ , the phase detector gain changes polarity, which causes the PLL to unlock. This experiment shows that the realizable hold range is only slightly less than the predicted one.

Let's now look at how the ADPLL reacts to phase steps. Check the Phi-Step radio button in the SIM dialog and enter  $d\phi = 90^\circ$ . Click the RUN button. After the phase step occurs, the output frequency is temporarily increased during a few reference cycles, but the loop settles to zero phase error very quickly (see Fig. 12.7).

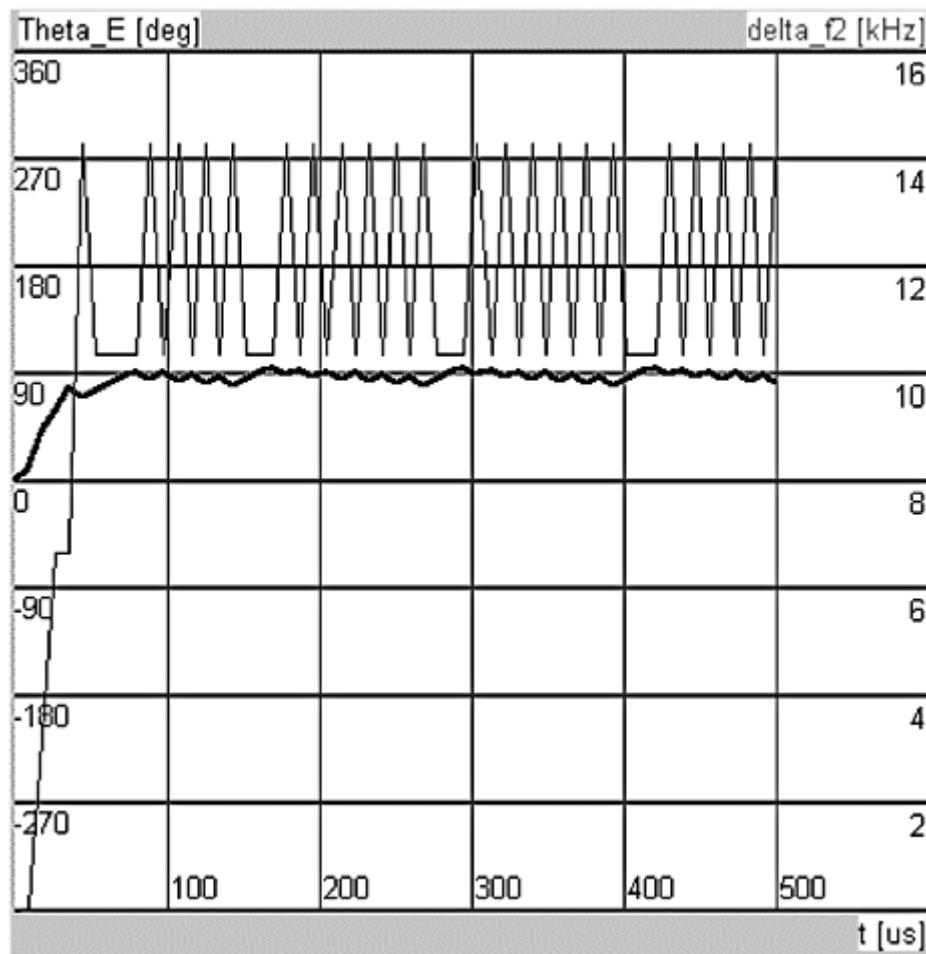
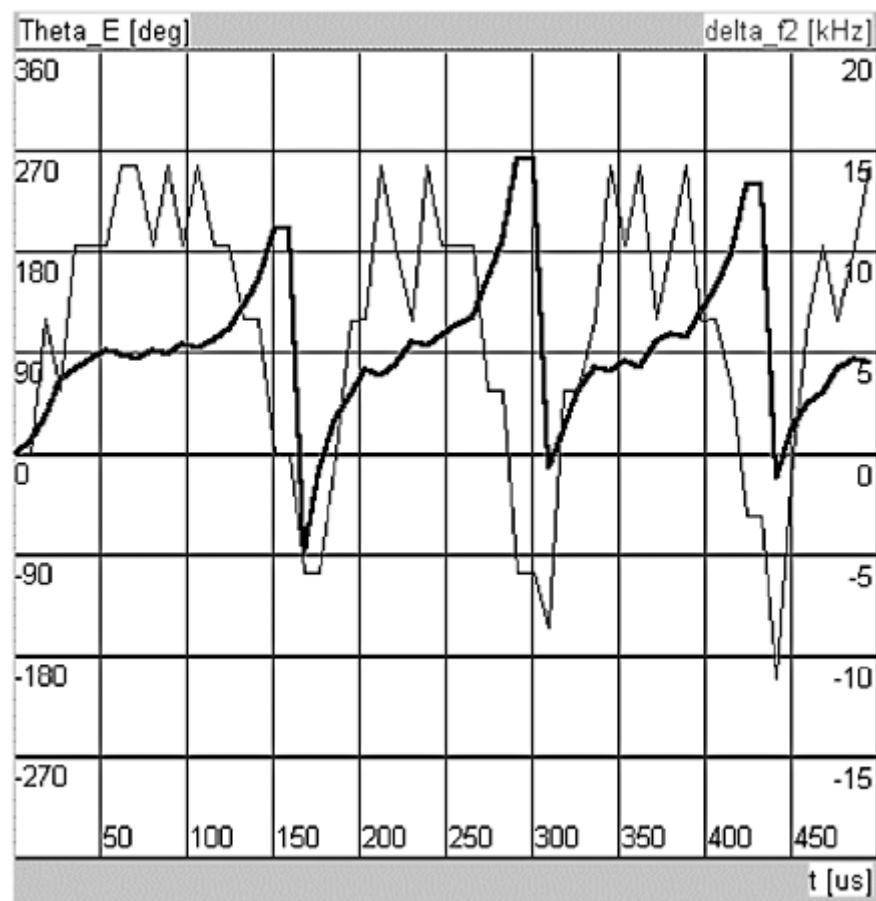
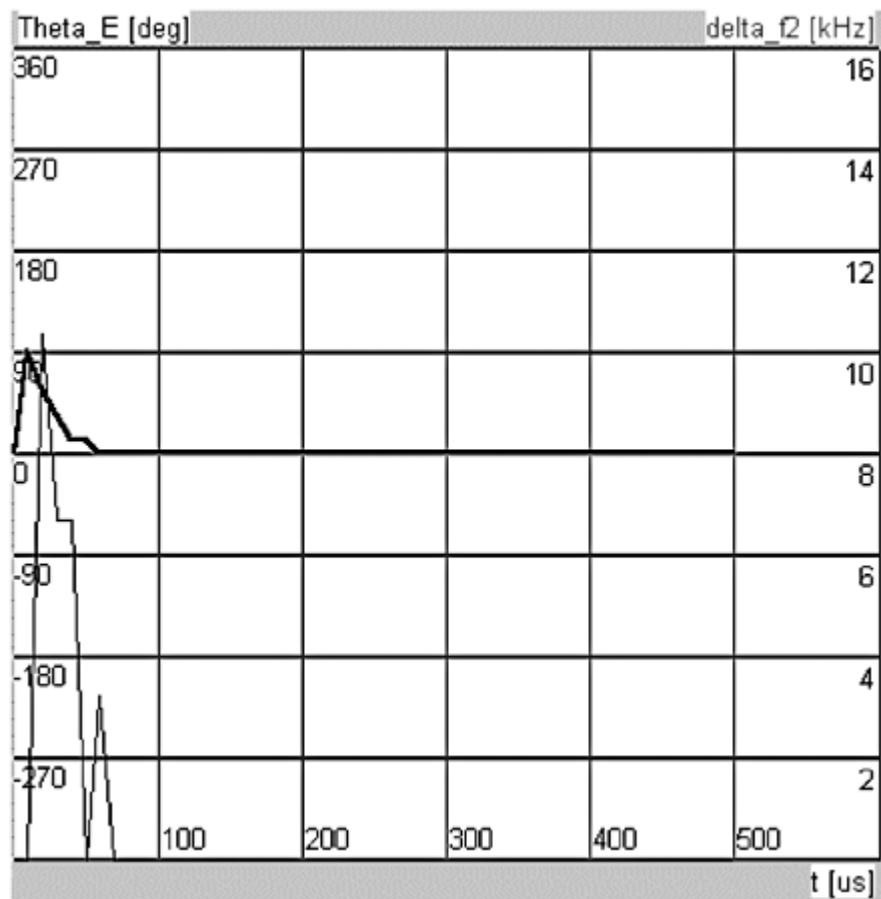


Figure 12.5 An ADPLL simulation. Response to frequency step  $df = 12$  kHz.

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.



**Figure 12.6** An ADPLL simulation. Response to frequency step  $df = 13$  kHz.



**Figure 12.7** An ADPLL simulation. Response to phase step  $\Delta\Phi = 90^\circ$ .

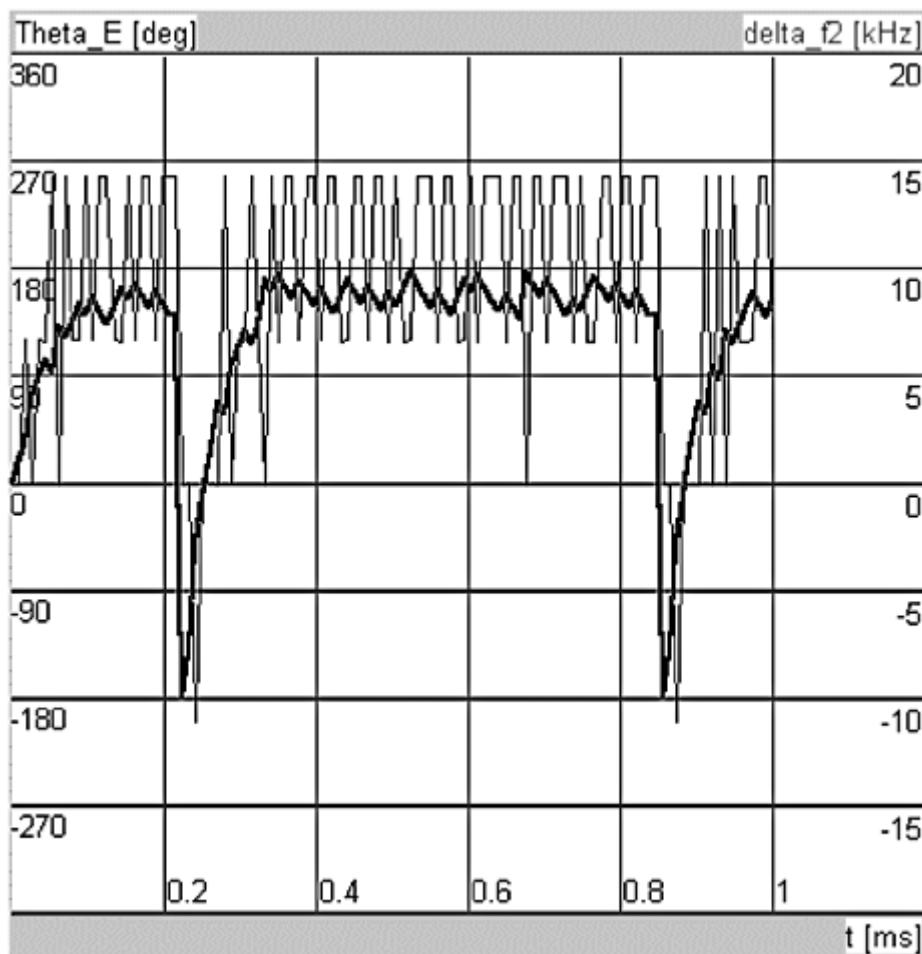
---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
 Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
 Any use is subject to the Terms of Use as given at the website.

**Case study 2.** *Dynamic performance of the ADPLL using the JK-flipflop PD.* Now we start another simulation and click the CONFIG speed button first. In the CONFIG dialog box, we check the JK-flipflop radio button. Enter a Center Frequency of  $f_0 = 100$  kHz again. Click the Done button to finish the CONFIG dialog and click the DESIGN speed button. In the ADPLL Params panel, enter the parameters for minimum ripple— $M = 16$ ,  $K = 8$ , and  $N = 8$ —and select the Calc Frequency Params button. The program predicts a hold range of  $f_H = 12.5$  kHz. According to ADPLL theory, the time constant of the loop becomes  $\tau = 40 \mu\text{s}$ . Click the Done button to exit the DESIGN dialog, and then click the SIM speed button. In the SIM dialog, enter a frequency step  $df = 11$  kHz and click the RUN button. The results of the simulation are shown in Fig. 12.8.

The loop locks, but the phase error nearly approaches the stability limit of  $180^\circ$ . If the frequency step is increased to 12 kHz, the loop is no longer able to maintain lock. It is not easy to read out the settling time accurately, but we recognize that the loop acquires lock after about  $120 \mu\text{s}$  which is roughly  $3\tau$ .

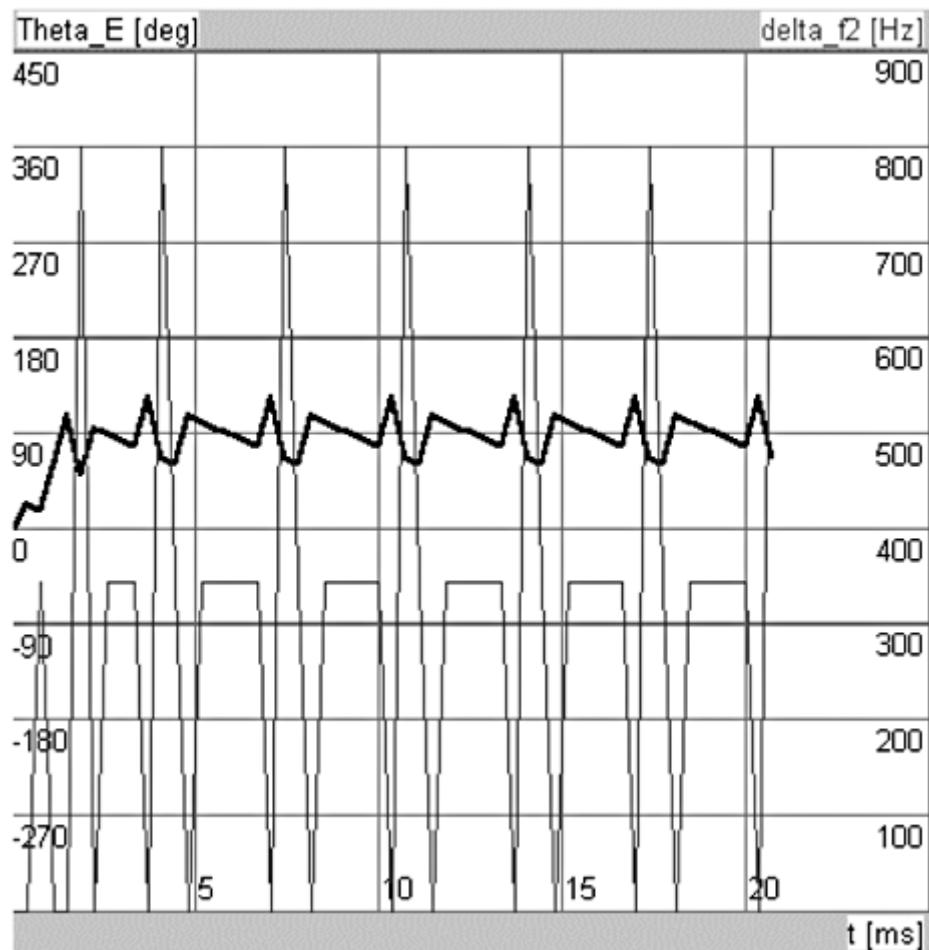
**Case study 3.** *Dynamic performance of the FSK decoder.* Let's now investigate the behavior of the FSK decoder we designed in Sec. 11.5.1. Start the simulation



**Figure 12.8** An ADPLL simulation. Response to frequency step  $df = 11$  kHz. Phase detector = JK-flipflop.

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.



**Figure 12.9** Response of the FSK decoder to a frequency step  $df = 300$  Hz.

by clicking the CONFIG speed button. In the PD Type panel, check the JK-flipflop radio button. In the Center Frequency panel, enter  $f_0 = 2400$  Hz. Click the Done button to close the CONFIG dialog. Select the DESIGN speed button to start the DESIGN dialog. In the DESIGN dialog, enter the parameters  $M = 16$ ,  $K = 8$ , and  $N = 4$ . Click the Calc Frequency Params button. The program calculates the expected hold range  $f_H = 600$  Hz. Close the DESIGN dialog by clicking the Done button. Enter simulation mode by clicking the SIM speed button now. In the SIM dialog, check the f-Step radio button and specify a frequency step  $df = 300$  Hz. Click the RUN button to start the simulation. The results are shown in Fig. 12.9.

Clearly, the ADPLL locks within a few reference cycles. The phase error is around  $90^\circ$ . There is quite a bit of ripple on the output frequency, which is not critical in this application. As Eq. (11.15) indicates, the frequency ripple could be reduced by increasing  $N$ . Doubling  $N$ , for example, would reduce the hold range by a factor of 2, which is not acceptable in this application. To maintain the hold range of 600 Hz,  $M$  would have to be doubled, too. The reader is encouraged to make their own trials on improving the frequency ripple of this ADPLL.

**Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.**

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

# The Software PLL (SPLL)

## The Hardware-Software Trade-off

In the age of microcontrollers and digital signal processors (DSPs), it is an obvious idea to implement a PLL system using software. When doing so, the functions of the PLL are performed by a computer program. The designer realizing a software PLL trades electronic components for microseconds of computation time. As the parts count for a hardware PLL increases with the level of sophistication, the number of computer instructions rises with the complexity of the required PLL algorithms.

Of course, the SPLL can compete with a hardware solution only if the required algorithms are executing fast enough on the hardware platform used to run the program. If a given algorithm performs too slowly on a relatively cheap micro-controller (one of the popular 8051 family, for example) and the designer is forced to resort to more powerful hardware (for instance, a DSP), a price tradeoff also comes into play. The high speed and low cost of available PLL ICs makes it difficult for the SPLL to compete with its hardware counterpart. Nevertheless, SPLLs can offer particular advantages, especially when computing power is already available.

When comparing SPLLs with hardware PLLs, we should recognize first that an LPLL or a DPLL actually is an analog computer that continuously performs some arithmetic operations. When a computer algorithm has to take over that job, it must replace these continuous operations using a discrete-time process. From our previous discussion of hardware PLLs, we know that every signal of such a system contains a fundamental frequency, which can be equal to its reference frequency  $f_1$  or twice that value. According to the sampling theorem, the algorithm of the SPLL must be executed two or even four times in each cycle of the reference signal. If the reference frequency is a modest 100 kHz, for example, the algorithm must execute 200,000 times per second in the most favorable case, which leaves not more than 5  $\mu$ s for one pass-through.

Today's microcontrollers easily work with clock frequencies up to the GHz region. For most microcontrollers, however, one instruction needs more—often much more—than one machine cycle to execute. There is a risk, therefore, that the microcontrollers on the lower end of the price scale fail to deliver the required computational throughput. Using DSPs instead brings us a big step forward, because they not only are fast with respect to clock frequency, but also offer Harvard-Plus and pipeline architecture.<sup>18</sup> Harvard architecture means that the DSP has physically separated data and program memories, and hence can fetch instructions and data within the same machine cycle. In even more sophisticated DSPs, the machine can fetch one instruction and several data words at the same time. The term "pipeline" implies that the arithmetic and logic units of the machine are fully decoupled, so that the DSP chip is able, for example, to perform one instruction fetch, some operand fetches (data fetches), one or more floating-point additions, one or more floating-point multiplications, one or more instructions decodings, one or more register-to-register operations, and perhaps even more in one single machine cycle. This greatly enhances computational throughput but results in higher cost, of course.

In the next section, we discuss the steps required to check the feasibility and economy of an SPLL realization.

## The Feasibility of an SPLL Design

An SPLL design offers the most degrees of freedom available in any one PLL design, because the SPLL can be tailored to perform similarly to an LPLL or a DPLL or to execute a function that none of these hardware variants is able to do. To check whether a software implementation can economically be justified, it is recommended you perform the following steps:

**Step 1. Definition of the SPLL algorithm.** The SPLL design procedure should start with the formal presentation of the algorithm(s) to be performed by the SPLL. Examples of such algorithms will be given in Sec. 13.3. For the moment, it is sufficient to write down these algorithms in symbolic form—that is, by algebraic and/or logic equations. Structograms are ideally suited to define the sequence of the operations, to describe conditional or unconditional program branchings, to describe loops that are repeatedly run through, and the like. Examples of structograms are also given in Sec. 13.3.

**Step 2. Definition of the language.** Having defined the algorithms, the language which will be used to encode them should be defined, at least tentatively. The programming effort is minimized when a high-level language such as C/C++, J/J++, or (object) PASCAL is used. If the program is required to finally run on a microcontroller, a language must be chosen for which a compiler is available. Manufacturers of microcontrollers or software houses mostly provide compilers for C/C++ and PASCAL. When the compiled assembly-language

program is available, the time required to execute it can be estimated. It should be noted that different assembler instructions may require different execution times.

Not every compiler is able to generate a time-efficient assembly code. If it is necessary to use a DSP, this point is even more important. When the DSP makes use of pipeline techniques,<sup>18,19</sup> the compiler must generate parallel assembly code—in other words, an assembler program where a number of different instructions are executed in any one instruction. In cases where efficient compiler programs are not available, the software designer could even be forced to write the program immediately in assembly code. With parallel-computing DSPs, this is not a simple task, however. Some manufacturers of DSP chips offer signal processing libraries written in assembly code, which can be used to perform most elementary signal-processing tasks—for example, digital filtering and the like.

Whatever language is used, the assembly code must be available to get an estimate of the approximate execution time of the algorithm(s).

**Step 3. Estimation of real-time bandwidth.** Having estimated the program execution time, the designer must calculate the real-time bandwidth of the SPLL system. If the execution time of the full SPLL algorithm is 50  $\mu$ s, for example, and two passes are required in one cycle of the reference signal, at least 100  $\mu$ s of computation time is needed in one reference period. Probably the microcontroller or whatever hardware is used will need some more time for time keeping, input/output operations, and the like; the real-time bandwidth is likely to fall well below 10 kHz in this example.

**Step 4. Real-time testing.** To check if the system performs as planned, the designer will have to implement a breadboard and test its system in “real time.” Only such a test can make sure the real system is not even slower than the designer imagines.

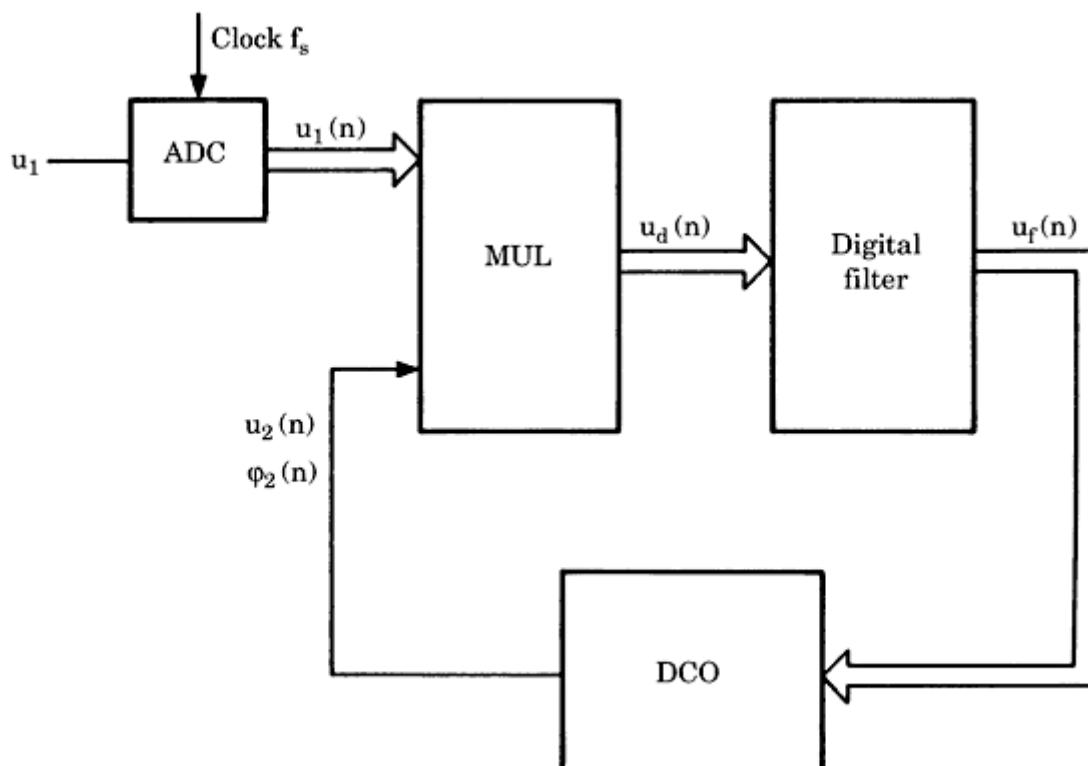
## SPLL Examples

Because every known LPLL, DPLL, or ADPLL system can be implemented by software, the number of variants becomes virtually unlimited. We therefore restrict ourselves to a few examples. The required algorithms for the SPLL will be described in great detail, so the reader should be able to adopt the methods to other SPLL realizations. The PLL simulation program delivered with the disk is a good example for SPLLs, because it demonstrates the ability of software to implement a great number of different linear and digital PLL configurations. You should be aware, however, that the simulation program does not represent a real-time system since it does not work with real signals or execute the algorithms in real time. Nevertheless, it uses a great deal of the algorithms described in the following sections.

## An LPPLL-like SPLL

We are going to develop an SPLL algorithm whose performance is similar to a hardware LPPLL. To derive the required SPLL algorithm, we plot a signal flow diagram which shows the arithmetic operations within the loop (Fig. 13.1). The input signal  $u_1$  is supposed to be an arbitrary analog signal—for instance, a sine wave. It is periodically sampled with the frequency  $f_s = 1/T$  by an analog-to-digital converter (ADC), where  $T$  is the sampling interval. Thus, samples are taken at times  $t = 0, T, 2T, \dots, nT$ .  $u_1(n)$  is the simplified notation for the input signal sampled at time  $t = nT$ —that is,  $u_1(n) = u_1(nT)$ . All other signals of the SPLL are sampled signals, too, and must be calculated at the sampling instants  $t = 0, T, 2T, \dots, nT$ . Consequently, all function blocks of the signal flow diagram are working synchronously with the ADC clock.

In Fig. 13.1, the signals shown by double lines are word signals. The output signal of the DCO, however, is a bit signal and is therefore represented by a single line. There are three function blocks in the signal flow diagram: a digital multiplier, a digital filter, and a DCO. (No down scaler is used in this configuration—in other words, for  $N = 1$ .) The multiplier is used as a phase detector and corresponds exactly with the already-known Nyquist rate PD discussed in Sec. 11.1.1; refer also to Fig. 11.2. Its output signal is denoted as  $u_d(n)$ . The digital filter serves as loop filter, while its output signal is  $u_f(n)$ . Finally, the DCO is supposed to generate a square-wave output signal  $u_2(t)$ , which is of course known only at the sampling instants. The sampled DCO output signal

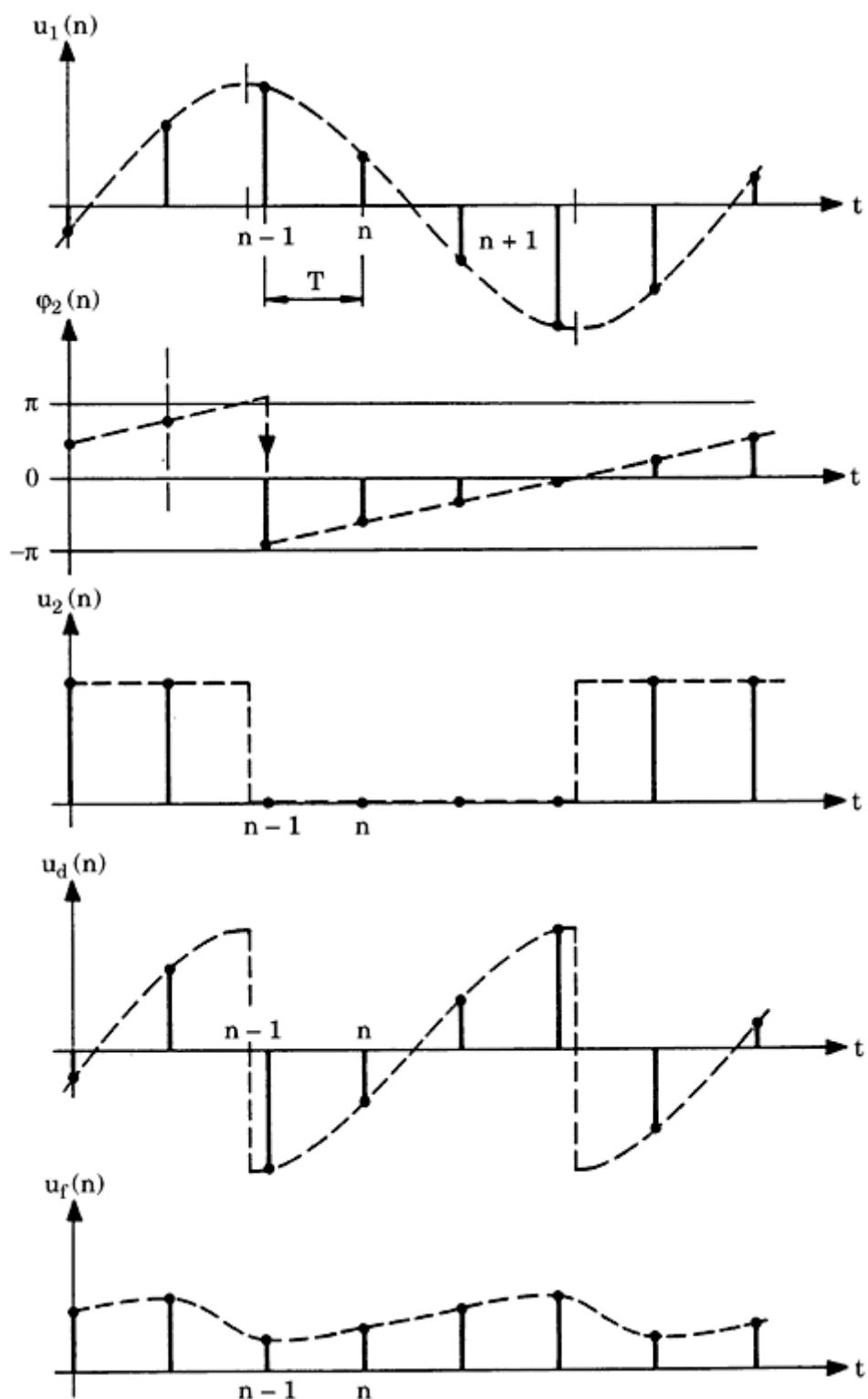


**Figure 13.1** A block diagram showing the arithmetic operations to be performed by an SPLL

whose performance is similar to the LPLL.

---

**Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.**



**Figure 13.2** A plot of the signals that must be calculated by the SPLL algorithm.

is denoted  $u_2(n)$ . As we will see, the DCO is not able to compute  $u_2(t)$  directly; this signal

must instead be calculated indirectly from the phase  $\varphi_2(t)$  of the DCO. If a VCO were used instead of the DCO, its instantaneous output angular frequency would be given by

$$\omega_2(t) = \omega_0 + K_0 u_f(t) \quad (13.1)$$

The continuous output signal  $u_2(t)$  then would be given by

$$u_2(t) = \text{rect}(\omega_2(t) \cdot t) \quad (13.2)$$

where *rect* denotes the square-wave function [refer to [Eq. \(2.10b\)](#)]. The total phase  $\varphi_2(t)$  of the VCO output signal then would be

$$\varphi_2(t) = \int \omega_2(t) dt = \omega_0 t + K_0 \int u_f(t) dt \quad (13.3)$$

Note that we dealt only with the differential phase  $\theta_2(t)$  hitherto, which corresponds to the second term on the right side of [Eq. \(13.3\)](#). Here, the total phase  $\varphi_2(t)$  is used to compute the instantaneous value of the DCO output signal  $u_2(t)$ . If we assign the values +1 and -1 to the square-wave signal, it follows from the definition of the square-wave function that  $u_2(t)$  is +1, when the phase  $\varphi_2(t)$  is either in the interval  $0 < \varphi_2 < \pi$  or in the interval  $2\pi < \varphi_2 < 3\pi$ , and so on. In all other cases,  $u_2(t) = -1$ . We will now adapt this computation scheme to the time-discrete case we are dealing with. When we know the digital filter output signal  $u_f(n)$  at sampling instant  $t = nT$  and assume furthermore that it stays constant during the time interval  $nT < t < (n+1)T$ , the total phase of the DCO output signal will change by an amount

$$\Delta\varphi_2 = [\omega_0 + K_0 u_f(n)]T \quad (13.4a)$$

in that interval. If the phase  $\varphi_2(n)$  at sampling instant  $t = nT$  were known, we would be able to extrapolate the total phase  $\varphi_2(n+1)$  at sampling instant  $t = (n+1)T$  from

$$\varphi_2(n+1) = \varphi_2(n) + [\omega_0 + K_0 u_f(n)]T \quad (13.4b)$$

This computation is possible because we can initialize the total phase with  $\varphi_2(0) = 0$  before the SPLL algorithm is started. Hence, we can extrapolate  $\varphi_2(1)$  at time  $t = 0 \cdot T$ ,  $\varphi_2(2)$  at  $t = 1 \cdot T$ , and so on. Given  $\varphi_2(n+1)$ , we can also extrapolate the value of  $u_2(n+1)$  at  $t = (n+1)T$ :

$$u_2(n+1) = 1 \quad \text{if } 2k\pi \leq \varphi_2(n+1) < (2k+1)\pi$$

or

$$u_2(n+1) = -1 \quad \text{if } (2k-1)\pi \leq \varphi_2(n+1) < 2k\pi$$

with  $k = \text{integer}$ .

The signals of our SPLL are depicted in [Fig. 13.2](#). The dashed lines represent continuous signals. The sampled signals are plotted as dots. Only the continuous signal  $u_1(t)$  really exists;

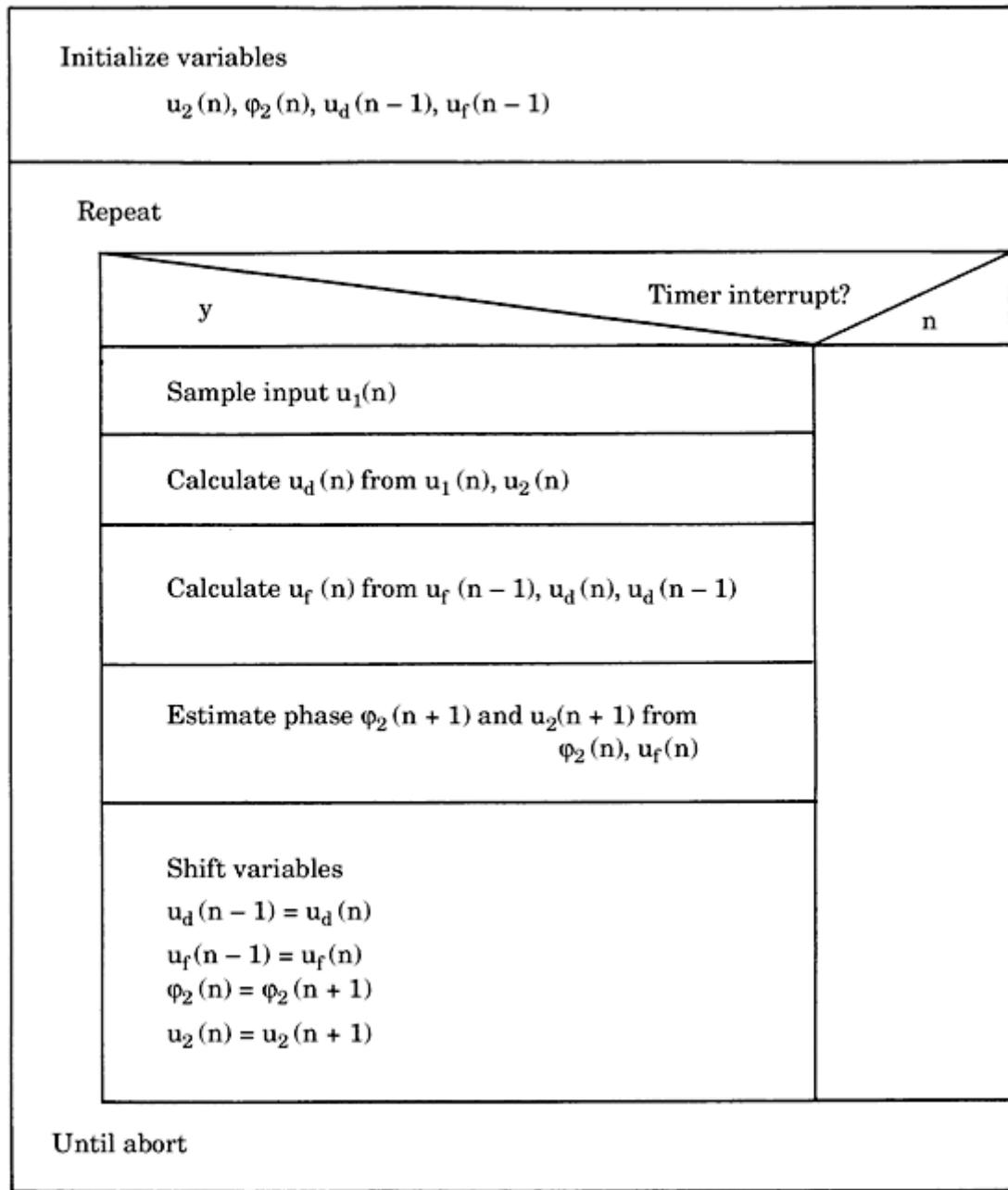
all others are fictive. The required algorithm is easily

derived from these waveforms. At a given sampling instant  $t = nT$ , the output signal  $u_d(n)$  of the multiplier has to be computed by

$$u_d(n) = K_d u_1(n) u_2(n)$$

where  $K_d$  is the gain of the phase detector. Given  $u_d(n)$ , a new sample of  $u_f(n)$  must be computed; the corresponding filter algorithm will be shown in the following. Given  $u_f(n)$ , the value of  $\varphi_2(n+1)$  at the next sampling instant is *extrapolated*. This enables us to extrapolate  $u_2(n+1)$  also. This value must be known, because we need  $u_2(n)$  in the following sampling instant to compute the next value of  $u_d(n)$ .

The SPLL algorithm is now represented symbolically in the structogram of [Fig. 13.3](#).



**Figure 13.3** A structogram defining the operations of the SPLL according to [Fig. 13.1](#).

When the algorithm is started, initial values are assigned to all relevant variables. The program enters an endless cycle thereafter—in other words, the algorithm within the box is repeatedly executed until the system is halted or switched off. It is assumed that the clock signal (refer to Fig. 13.1) periodically generates interrupts in the microcontroller or whatever hardware is used. Thus, interrupt requests show up at time instants  $t = T, 2T, \dots, nT$ . As soon as the interrupt is recognized by the hardware, the SPLL algorithm is executed. It starts with the acquisition of a sample of the input signal  $u_1(t)$ . The next three statements of the structogram correspond with the computation scheme already described. Finally, when all variables of the SPLL have been updated, they must be delayed (or shifted in time). The variable  $u_d(n-1)$  is overwritten by the value  $u_d(n)$ , which means that the “new” value of  $u_d(n)$  computed in this cycle will be the “old” value  $u_d(n-1)$  in the next cycle. The same holds true for all other variables.

Knowing what has to be calculated in every step, we can develop the algorithm in mathematical terms. The full procedure is listed in Fig. 13.4. First, all relevant variables are initialized with 0. Depending on the particular application, other values can be appropriate. The operation of the multiplier is trivial. The next statement is the digital filter algorithm:

$$u_f(n) = -a_1 u_f(n-1) + b_0 u_d(n) + b_1 u_d(n-1) \quad (13.5)$$

This is the recursion of a first-order digital filter. As pointed out in Sec. 11.1.2, an analog filter is described by its transfer function

$$F(s) = \frac{U_f(s)}{U_d(s)} \quad (13.6)$$

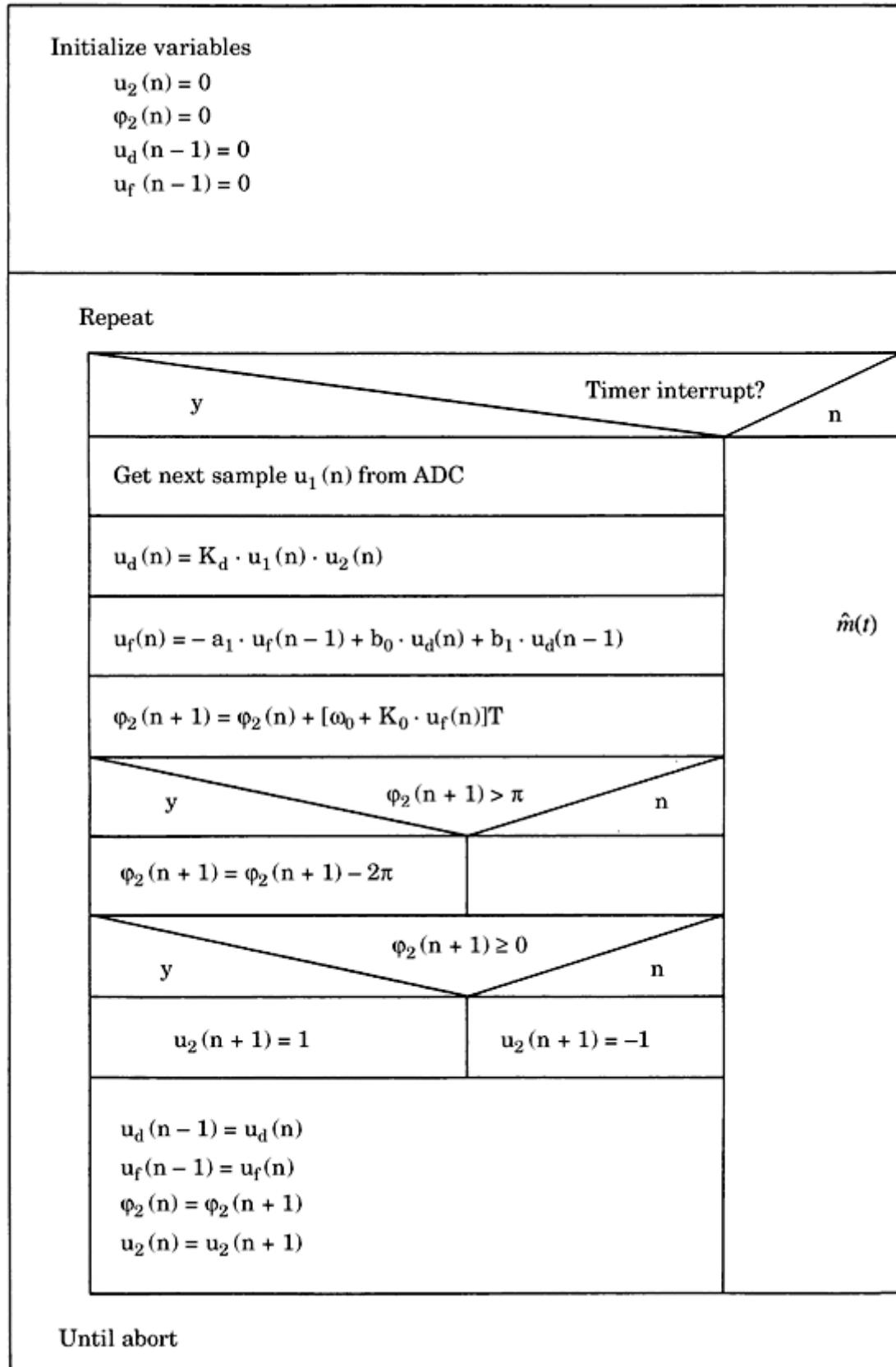
where  $s$  is the Laplace operator and  $U_f(s)$  and  $U_d(s)$  are the Laplace transforms of the continuous signals  $u_f(t)$  and  $u_d(t)$ , respectively. To get a digital filter performing nearly the same function, we usually transform  $F(s)$  into the  $z$ -domain

$$F(z) = \frac{U_f(z)}{U_d(z)} \quad (13.7)$$

where  $z$  now is the  $z$ -operator. A number of transforms<sup>12,13,14,19</sup> can be used to convert  $F(s)$  into  $F(z)$ . The type most frequently used is called the bilinear  $z$ -transform. It will be covered in some more detail in App. C. Before the digital filter is designed, we start with the definition of the (fictive) analog filter. Because we know that the active PI filter offers the best PLL performance, we assume that  $F(s)$  is the transfer function of the active PI filter [refer also to Eq. (2.31a)]. Using the bilinear  $z$ -transform, we get

$$F(z) = \frac{b_0 + b_1 z^{-1}}{1 + a_1 z^{-1}} \quad (13.8)$$

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.



**Figure 13.4** A structogram showing the algorithm to be performed by the SPLL in each sampling interval.

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

where the filter coefficients are given by

$$a_1 = -1$$

$$b_0 = \frac{T}{2\tau_1} \left[ 1 + \frac{1}{\tan(T/2\tau_2)} \right]$$

$$b_1 = \frac{T}{2\tau_1} \left[ 1 - \frac{1}{\tan(T/2\tau_2)} \right]$$

and  $T$  is the sampling interval. Transforming [Eq. \(13.8\)](#) back into time domain, we get the recursion

$$u_f(n) = -a_1 u_f(n-1) + b_0 u_d(n) + b_1 u_d(n-1) \quad (13.9)$$

which is also listed in the structogram of [Fig. 13.4](#). Using [Eq. \(13.4b\)](#), the total phase of the DCO output signal at the next sampling instant will be

$$\varphi_2(n+1) = \varphi_2(n) + [\omega_0 + K_0 u_f(n)]T$$

When the algorithm is executed over an extended period of time, the values of  $\varphi_2(n+1)$  will become very large and could soon exceed the allowable range of a floating number in the processor used. To avoid arithmetic overflow,  $\varphi_2$  is limited to the range  $-\pi < \varphi_2 < \pi$ . Whenever the computed value of  $\varphi_2(n+1)$  exceeds  $\pi$ ,  $2\pi$  is subtracted to confine it to that range. Now the value of  $u_2(n+1)$  is easily computed by checking the sign of the range-limited total phase. If  $\varphi_2(n+1) \geq 0$ ,  $u_2(n+1) = 1$ ; otherwise,  $u_2(n+1) = -1$ . Finally, the calculated values of  $u_d(n)$  and others are delayed by one sampling interval:

$$u_d(n-1) \leftarrow u_d(n)$$

and so on.

As we stated in [Sec. 10.3](#), when simulating the PLL on the PC, the sampling rate  $f_s$  for this SPLL algorithm must be chosen to be at least four times the reference frequency in order to avoid aliasing of signal spectra.

## A DPLL-like SPLL

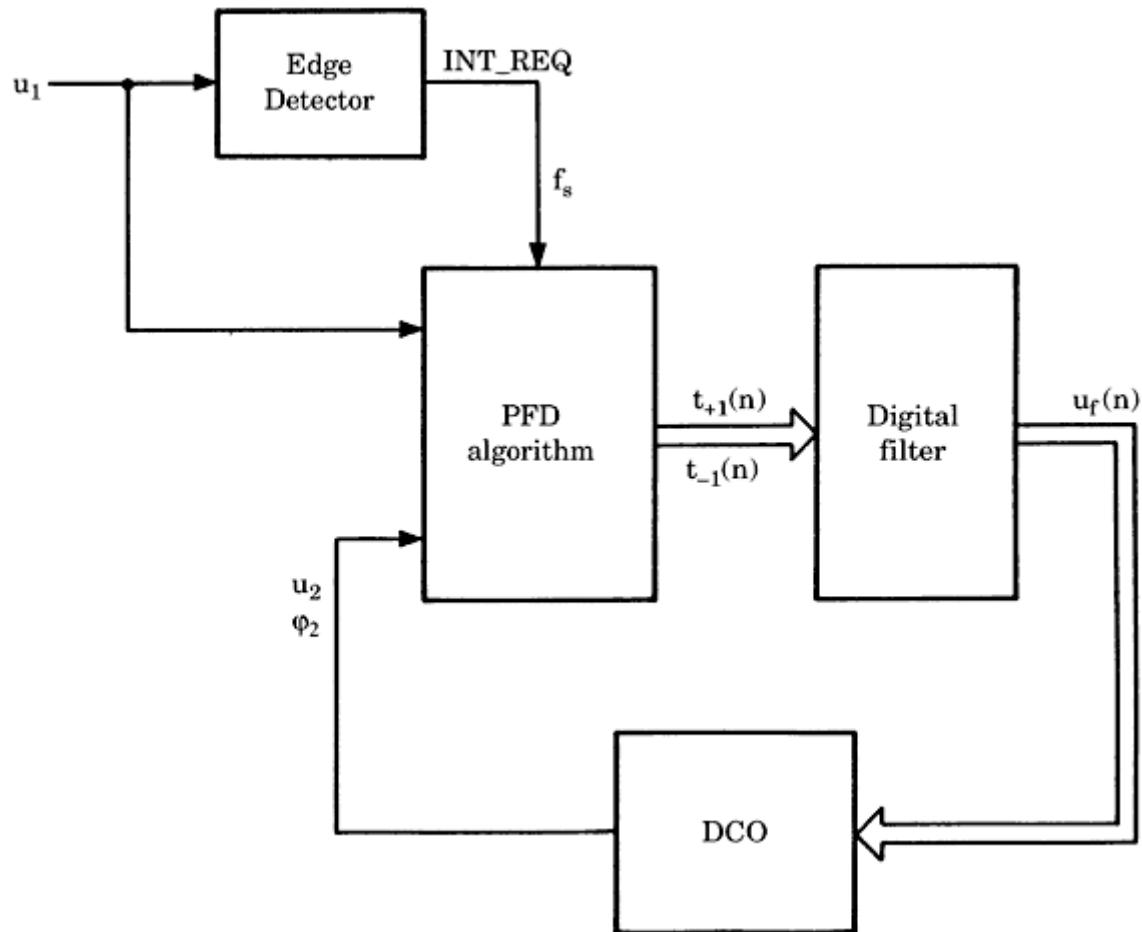
When the input signal  $u_1$  of a PLL is a binary signal, it is more adequate to implement an SPLL that performs like a DPLL. We develop an algorithm now that performs like the DPLL using the phase-frequency detector and a passive lead-lag filter. Though the mathematical and logical operations within such a DPLL seem simpler compared with an LPLL, it turns out that the algorithm for the corresponding SPLL becomes much more complicated.

Let us again represent the required functions by a signal flow diagram ([Fig. 13.5](#)). It essentially consists of three functional blocks: a phase frequency detector (PFD) algorithm, a

digital filter, and a DCO. (No down scaler is used in this configuration, such as  $N = 1$ .) The digital filter is required to operate like

---

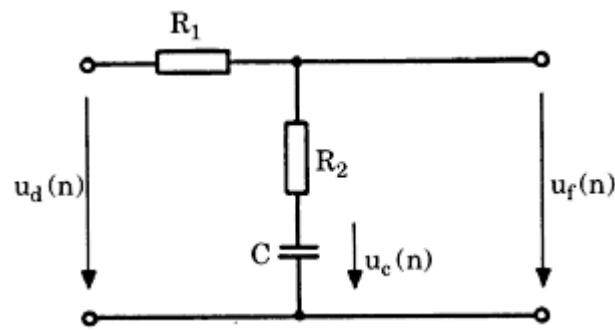
Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.



**Figure 13.5** A block diagram showing the arithmetic and logic operations to be performed by an SPLL whose performance is similar to the DPLL.

the passive lead-lag filter in a DPLL, which is once more shown in Fig. 13.6. Before going into detail about the last two figures, we should consider the signals of this SPLL (Fig. 13.7). The only signal that physically exists—at least at the beginning—is the input signal  $u_1(t)$ , a square whose frequency can vary within the frequency range of the DCO. The (fictive) output signal  $u_2(t)$  of the PLL would be a square wave, too.

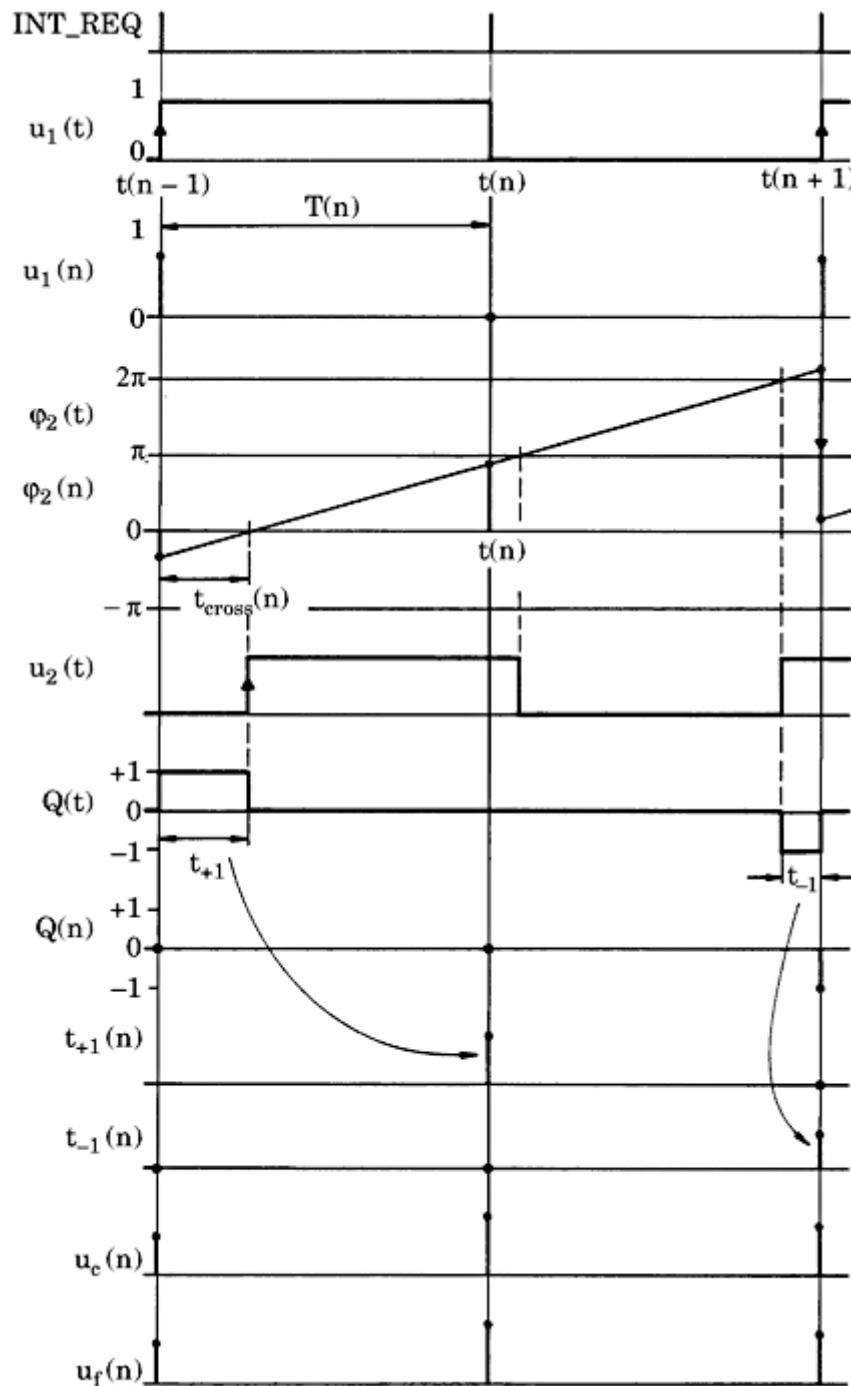
As we know from Sec. 2.4.4.1, the logic state  $Q$  of the PFD depends on the positive edges of these two signals (or from the negative edges, whatever definition is made). When the PLL has settled to a steady state, the signals  $u_1(t)$  and  $u_2(t)$  are nearly in phase. The output  $Q$  of the PFD is then in the 0 state most of the



**Figure 13.6** A schematic of the passive lead-lag loop filter. This drawing is used to define the variables of the loop filter.

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.



**Figure 13.7** A plot of the signals that must be calculated by the SPLL algorithm.

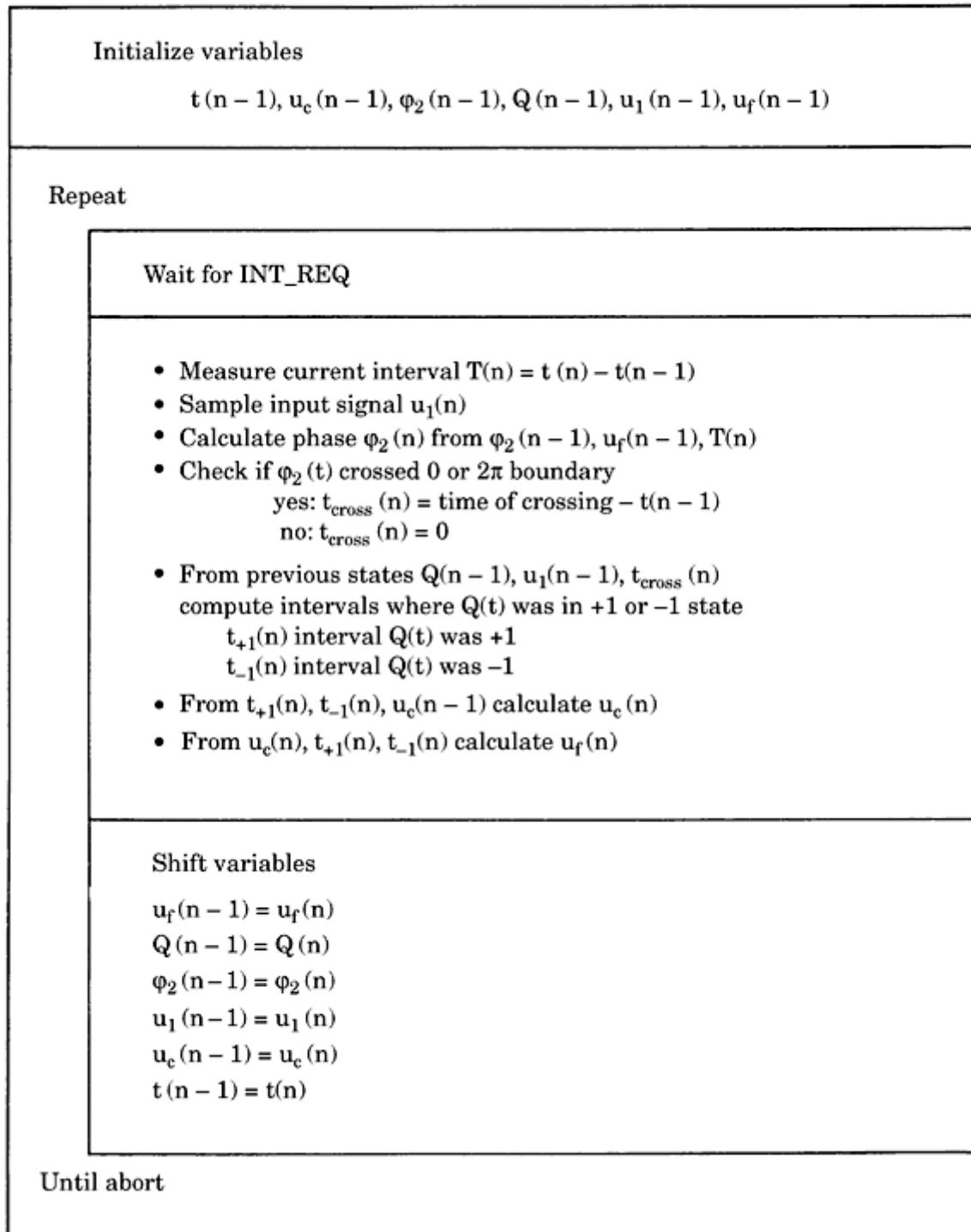
time. Should the output frequency of the DCO drift away, the PFD would generate correction pulses—in other words,  $Q$  would become  $+1$  or  $-1$  for a very short time. The width of the correction pulses is mostly less than 1/1000 of one period of the reference signal. If we tried to detect the edges of  $u_1(t)$  and  $u_2(t)$  by sampling these signals, the sampling frequency would have to be at least 1000 times the reference frequency, which is highly unrealistic.

Another scheme must therefore be used to detect the instants where the states of  $u_1$  and  $u_2$  are changing. Because we need to know the times where  $u_1$  and  $u_2$  are switching from low to high, we use the (positive and negative) edges

of  $u_1(t)$  to generate interrupt requests to the computer (refer to the signal INT REQ in Figs. 13.5 and 13.7). The computer is supposed to have a timer/counter chip such as the Intel 8253<sup>32</sup> or the AMD 9513.<sup>33</sup> As soon as the interrupt is recognized, a “time stamp” is taken—thus, the time when the interrupt occurred is stored. The instants where interrupts have been detected are called  $t(0)$ ,  $t(1)$ ,  $\dots$   $t(n)$ , and so on. Three of them are marked at the top of Fig. 13.7. Before the SPLL algorithm can be discussed, we must define a number of signals (refer to Fig. 13.7).

- $u_1(n)$  is the sampled version of the continuous reference signal  $u_1(t)$  immediately after occurrence of the interrupt request. At time  $t(n - 1)$ , for example,  $u_1(n - 1) = 1$ , and at time  $t(n)$ ,  $u_1(n) = 0$ .
- $\varphi_2(t)$  is the (fictive) continuous phase of the DCO output signal.
- $\varphi_2(n)$  is the sampled version of  $\varphi_2(t)$ . Of course, the samples are also taken at the instants where an interrupt occurred.
- $u_2(t)$  is the (fictive) continuous output signal of the DCO. It will be calculated from the phase  $\varphi_2(t)$ , as in the example in Sec. 13.3.1.
- $Q(t)$  is the (fictive) continuous output signal (or state) of the PFD. It can have the values  $-1$ ,  $0$ , or  $1$ .
- $Q(n)$  is a sampled version of  $Q(t)$  and is defined to be the state of the PFD just prior to the occurrence of the interrupt at time  $t(n)$ . For example,  $Q(n - 1)$  has the value  $0$ , because  $Q(t)$  was in the  $0$  state before the interrupt at  $t = t(n - 1)$  was issued.
- $T(n)$  is defined to be the time interval between the time of the most recent interrupt  $t(n)$  and the time of the preceding interrupt at  $t = t(n - 1)$ ; thus  $T(n) = t(n) - t(n - 1)$ . When  $Q(t)$  is in the  $+1$  state in a fraction of the  $T(n)$  interval, the corresponding duration is stored in the variable  $t_{+1}(n)$ , as shown by the arrow in Fig. 13.7. When  $Q(t)$  is in the  $-1$  state in a fraction of the  $T(n)$  interval, however, the corresponding duration is stored in the variable  $t_{-1}(n)$ ; this is indicated by another arrow on Fig. 13.7.
- Finally,  $u_c(n)$  is used to denote the signal on the (fictive) capacitor  $C$  in the schematic of Fig. 13.6;
- $u_f(n)$  is used to denote the sampled output signal of the digital filter in Fig. 13.5. With reference to Fig. 13.6,  $u_f(n)$  is nearly identical with  $u_c(n)$  but can slightly differ when “current” flows in the (fictive) resistor  $R_2$ .

The enumeration of that large set of variables has been quite cumbersome, but the elaboration of the algorithms will be even more fatiguing. The structogram of Fig. 13.8 shows what must be done on every interrupt request. The signals appearing in the algorithm are shown in Fig. 13.7. The uppermost portion of the SPLL algorithm is trivial and lists the initialization of some variables. As in the previous example, the program then enters an endless loop where it waits for the next interrupt. When the interrupt has been detected, the time lapsed since the last interrupt was taken  $T(n) = t(n) - t(n - 1)$ . Next, the



**Figure 13.8** A structogram defining the arithmetic and logic operations within the SPLL of Fig. 13.5.

current value of the reference signal  $u_1(t)$  is sampled,  $u_1(n) = u_1(t)$ . This is necessary because we need to know whether we are in the positive or negative half-cycle of the square wave  $u_1$

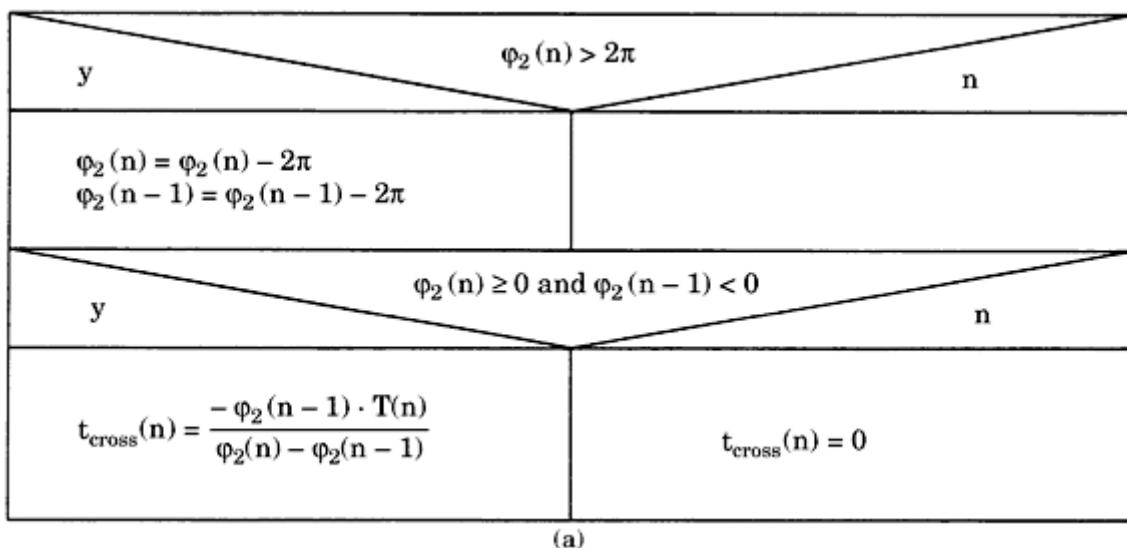
( $t$ ). We assume that the current time  $t$  is  $t(n)$  right now, which corresponds to the second interrupt request shown in the middle of Fig. 13.7. In contrast to the previous SPLL example, we do not know the value of the phase  $\varphi_2(t)$  at that time! The reason for this is simple: At time  $t = t(n - 1)$ , the value of

the digital filter output signal  $u_f(n - 1)$  can be calculated, and consequently we also know the instantaneous (angular) frequency  $\omega_2(n - 1)$  of the DCO. But since we did not yet know at time  $t = t(n - 1)$  how long the duration of the following half-cycle of  $u_1(t)$  would be, we could not extrapolate  $\varphi_2(n)$  but had to postpone that until  $t = t(n)$ . Only now, at  $t = t(n)$ , are we able to compute  $\varphi_2(n)$  from

$$\varphi_2(n) = \varphi_2(n - 1) + [\omega_0 + K_0 u_f(n - 1)] T(n) \quad (13.10)$$

Note that the phase  $\varphi_2(n - 1)$  at time  $t = t(n - 1)$  was known, because the phase signal is computed recursively and was initialized with  $\varphi_2(0) = 0$  at  $t = 0$ . Next, we must determine whether or not the (fictive) signal  $u_2(t)$  showed up a positive edge in the interval  $T(n)$ . This is the case when the continuous phase signal  $\varphi_2(t)$  “crossed” the value 0 or  $2\pi$  during interval  $T(n)$ ; this is sketched in the waveforms of Fig. 13.7. (The attentive reader will have noted that positive edges also would occur at a phase crossing with  $4\pi$ ,  $6\pi$ , and so on. As will be shown later, we periodically reduce the total phase by  $2\pi$  whenever it becomes larger than  $2\pi$ . This is necessary to avoid arithmetic overflow in the computer.) When the phase crosses such a boundary, the corresponding time [that is, the time interval from  $t(n - 1)$  to the crossing] is stored in the variable  $t_{\text{cross}}(n)$ . When no crossing was detected,  $t_{\text{cross}}(n)$  is set to 0. The algorithm for the computation of  $t_{\text{cross}}(n)$  is indicated in the structogram of Fig. 13.9a. It starts with the “normalization” of the phase signal  $\varphi_2(t)$ , as noted earlier.

We are ready now to compute the state  $Q(t)$  of the PFD during the interval  $T(n)$ . The signal  $Q(t)$  depends on a number of other variables. First of all, the state of  $Q(n - 1)$  prior to time  $t = t(n - 1)$  must be known. If, as sketched in Fig. 13.7,  $Q(n - 1)$  was 0 and  $u_1(t)$  made a positive transition at  $t = t(n - 1)$ ,  $Q(t)$  goes into the +1 state. When  $u_2(t)$  also makes a positive transition thereafter,



**Figure 13.9** Detailed structograms of the algorithms to be performed by the SPLL of Fig.

[13.5.](#) (a) An algorithm to determine the variable  $t_{\text{cross}}(n)$  [time when output phase  $\varphi_2(t)$  crosses the boundary of 0 or  $2\pi$ ]. (b) An algorithm for the PFD. (c) An algorithm for the digital filter.

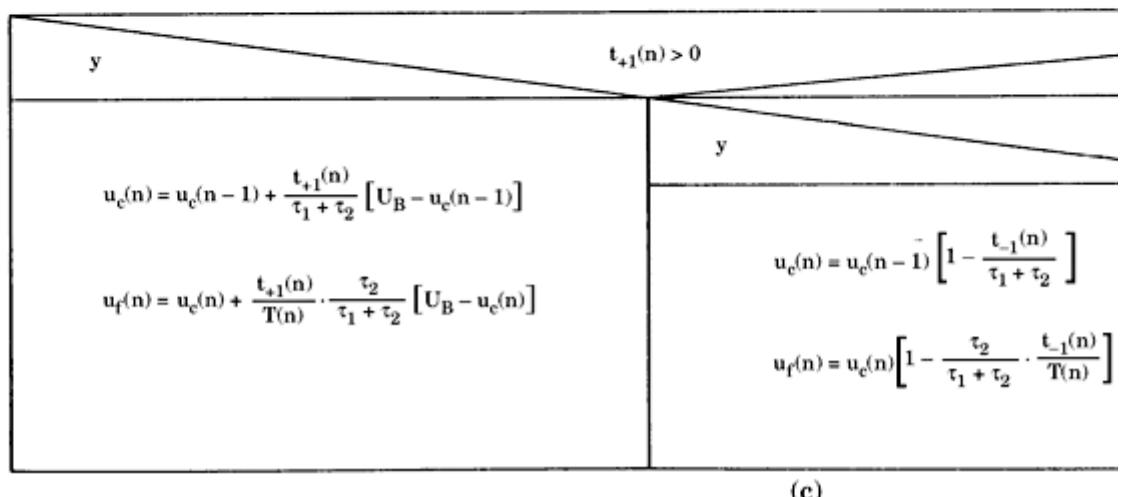
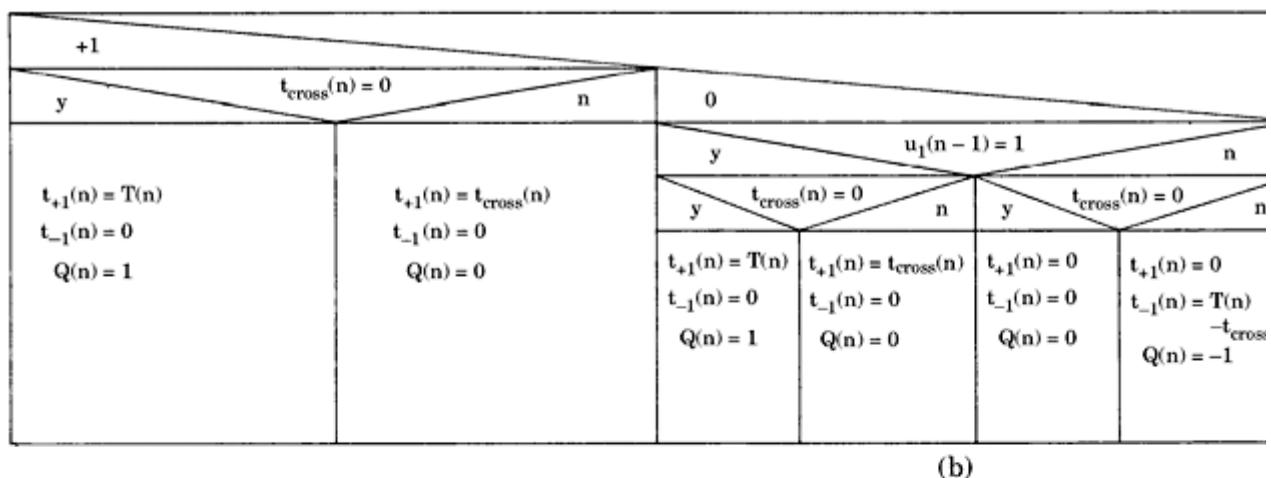


Figure 13.9 (Continued)

$Q(t)$  goes back to the 0 state. If  $Q(n - 1)$  had already been in the +1 state at  $t = t(n - 1)$ , however, it could not have changed its state on the positive edge of  $u_1(t)$ . The behavior of the PFD is therefore case-sensitive; the algorithm in Fig. 13.9b demonstrates that as many as nine different cases are possible. This algorithm determines the values of  $t_{+1}(n)$  and  $t_{-1}(n)$  and also computes the state of  $Q$  at the end of the  $T(n)$  interval. This state will be used as the initial condition  $Q(n - 1)$  in the next interrupt service.

When it turns out that  $t_{+1}(n)$  is greater than zero, this means that the “supply voltage  $U_B$ ” must be applied during interval  $t_{+1}(n)$  to the RC filter of Fig. 13.6. When  $t_{-1}(n)$  is nonzero, however, the “capacitor”  $C$  would have to be discharged toward ground during the interval  $t_{-1}(n)$ . The digital filter algorithm in Fig. 13.9c explains how the voltage  $u_c(n)$  on capacitor  $C$  must be computed from the previous value  $u_c(n - 1)$ . If no current flowed into or out of the capacitor  $C$  (Fig. 13.6), the output signal  $u_f(n)$  would be identical with capacitor voltage  $u_c(n)$ . In the intervals where current flows, however,  $u_f(n)$  can be higher or lower than  $u_c(n)$ , depending on the polarity of the current. Because  $u_f(t)$  is nonconstant in the interval  $T(n)$ , we define  $u_f(n)$  to be the average of  $u_f(t)$  in the interval  $t(n - 1) \leq t < t(n)$ . This yields the expression listed in Fig. 13.9c. When deriving the equations in this algorithm, it was assumed that the duration of a  $T(n)$  cycle (half a cycle of the reference signal) is much smaller than the filter time constant  $\tau_1$  in Fig. 13.6. Under this condition, the current flowing into or out from capacitor  $C$  remains constant during the charging or discharging intervals. This assumption leads to simpler expressions for  $u_c(n)$  and  $u_f(n)$ .

The algorithm used to compute the filter output  $u_f(n)$  differs considerably from conventional digital filter algorithms. In a classical filter algorithm, the sample  $u_f(n)$  of the output signal is calculated from a number of delayed samples of the output signal and from a number of delayed samples of its input signal. This scheme does not apply, however, to the current example, because the input signal of this circuit is applied only during a *fraction of the sampling interval*. The input is “floating” in the remaining time. Hence, the output signal must be calculated like the output of an analog filter, where the input is applied *continuously*.

All computations of one interrupt service are done now. Because most of the computed samples at  $t = t(n)$  will be used as starting values in the next interrupt service, they must be shifted in time. This is indicated in the bottom of the structogram in Fig. 13.8. Finally, the structogram of Fig. 13.10 lists the full algorithm in mathematical statements. To avoid overloading the graph, the algorithms for  $t_{\text{cross}}(n)$ , for the PFD, and for the digital filter are shown separately (Fig. 13.9a to c). Note that the only signal that physically exists hitherto is the reference signal  $u_1(t)$ . A realistic SPLL system should also have one or more real output signals. When the SPLL is used as an FSK decoder, for example, the demodulated signal is represented by  $u_f(n)$ , which is directly computed in every interrupt service. It would be quite simple to apply  $u_f(n)$  to an output port of the microcontroller whenever it is computed. The situation would become more troublesome if the SPLL were requested to deliver the continuous DCO output.

```
Set all filter parameters  
( $\tau_1, \tau_2, U_B, \omega_0, K_0, \dots$ )  
  
Initialize variables  
 $t(n-1) = 0$   
 $u_c(n-1) = 0$   
 $\varphi_2(n-1) = 0$   
 $Q(n-1) = 0$   
 $u_1(n-1) = 0$   
 $u_f(n-1) = 0$   
  
REPEAT  
  
    WAIT FOR INTERRUPT  
  
     $T(n) = t(n) - t(n-1)$   
  
    Sample input signal  $u_1(n)$   
  
     $\varphi_2(n) = \varphi_2(n-1) + [\omega_0 + K_0 \cdot u_f(n-1)] T(n)$   
  
    Algorithm for  $t_{cross}(n)$  (Fig.13-9a)  
  
    Algorithm for PFD (Fig.13-9b)  
  
    Algorithm for digital filter (Fig.13-9c)  
  
    Shift variables  
         $Q(n-1) = Q(n)$   
         $\varphi_2(n-1) = \varphi_2(n)$   
         $u_1(n-1) = u_1(n)$   
         $u_c(n-1) = u_c(n)$   
         $u_f(n-1) = u_f(n)$   
         $t(n-1) = t(n)$   
  
UNTIL abort
```

**Figure 13.10** A structogram showing the complete algorithm of the SPLL of [Fig. 13.5](#).

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

signal  $u_2(t)$ . As we see from the waveforms in Fig. 13.7, the SPLL algorithm determines at time  $t = t(n)$  when this signal made its last transition—thus, it only knows what happened in the past. To output a real-time signal, the SPLL algorithm could extrapolate at time  $t(n)$  how much time should lapse until the next transient of  $u_2$ . The corresponding time delay could then be loaded into a timer, which causes another interrupt request when timed out. The corresponding interrupt routine would finally set a bit of an output port with the current state of the  $u_2$  signal. This simple example demonstrates that it can become quite cumbersome if only a simple hardware device must be replaced by software.

### A note on ADPLL-like SPLLs

There is no question that all hardware ADPLLs can easily be implemented by software. Every hardware ADPLL operates under the control of one or several clock signals. On each clock pulse, a number of arithmetic and/or logic operations are performed. In the corresponding SPLL algorithm, these clock signals must be replaced by interrupt requests, and the interrupt service routines must perform the operations triggered by the clock in the hardware ADPLL. It is a relatively simple task to implement the algorithm of the popular 74HC/HCT297 IC (discussed in Sec. 11.3) by software. The author has built in such an algorithm in the simulation program distributed with the disk. Assume that the 74HC/HCT297 circuit is used for clock signal recovery in a modem operating at a baud rate of 9600 bauds. This circuit would then operate with a center frequency of  $f_0 = 9600$  Hz. The  $K$  modulus of the  $K$  counter cannot be smaller than  $K = 8$ , as pointed out in Sec. 11.3. To get minimum ripple,  $M$  (the multiplier of the  $K$  clock) would be chosen  $M = 4K = 32$ . Moreover, the assumption  $M = 2N$  is made whenever possible, where  $N$  is the scale factor of the external  $\div N$  counter. Consequently, the  $K$  counter and the ID counter would both operate at a frequency of  $M \cdot f_0 = 307,200$  Hz. If the ADPLL algorithm must be implemented by software, the corresponding interrupt service routine must execute more than 300,000 times in a second, and one single pass of the routine should take less than 3  $\mu$ s, including the operations required to service the interrupt request. By the present state of the art, this is out of reach of simpler microcontrollers such as the popular 8051 family.<sup>34</sup> Unfortunately, most hardware ADPLLs use clock signals whose frequency is a multiple of the center frequency; hence, their realization by software stays restricted to low-frequency applications. Of course, the upper frequency limit can be extended if the more powerful DSPs are used as hardware platforms.

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

# The PLL in Communications

## Types of Communications: Baseband and Bandpass

In this chapter, we will discuss applications of the PLL in the domain of data communication. Since many different kinds of communications exist, we will first have a look at the most important variants. To begin with, analog or digital data can be transmitted over a data link. Historically, the PLL was developed to be used in the analog domain: the inventor Henri de Bellescize<sup>22</sup> designed a vacuum tube-based synchronous demodulator for an AM receiver. The first important application of the PLL was in the recovery of the color subcarrier in television receivers around 1950.<sup>2</sup>

In recent years, digital communications have become increasingly important, even in the classical analog domains such as telephone, radio, and television. Because synchronization is an extremely important task whenever digital data is transmitted, the PLL and related circuits find widespread applications.

Analog and digital signals can be transmitted either as baseband signals or by modulation of a carrier; in the latter case, we are referring to bandpass modulation. If analog signals are communicated in the baseband, there is no need for synchronization, hence this is not an issue for the PLL. PLLs and related circuits come into play, however, when analog signals are modulating a high-frequency carrier. For analog signals, the classical modulation schemes are still *amplitude modulation (AM)*, *frequency modulation (FM)*, and *phase modulation (PM)*.

Historically, *AM* is the oldest modulation scheme. To modulate an analog signal (for example, voice, music, and measurements such as temperature or humidity) onto a carrier, an analog multiplier can be used. The multiplication can also be performed by a digital multiplier; this operation may also be executed by software on a suitable platform, say, a microcontroller. If the multiplication is done digitally, the modulated carrier signal must be converted into analog form by a DAC, of course. The demodulation of AM signals can be carried out by an LPLL circuit similar to that shown in Fig. 4.7. When a linear PLL

locks onto an AM modulated carrier whose frequency is identical with the center frequency of the PLL, there is a phase difference of  $90^\circ$  between the carrier ( $u_1$ ) and the VCO output signal ( $u_2$ ). The  $90^\circ$  phase shifter in Fig. 4.7 delivers an output signal  $u_1'$ , which is in phase with the reconstructed carrier  $u_2$ , and hence can be used to synchronously demodulate the signal using a four-quadrant multiplier. A lowpass filter eliminates the unwanted high-frequency component at about twice the carrier frequency. Of course, the Schmitt trigger in Fig. 4.7 must be removed from this circuit when it is applied as an AM demodulator. The bandwidth of the lowpass filter must be matched to the bandwidth of the information signal. If the information signal has a bandwidth of 4.5 kHz, for example (as in AM radio), the lowpass filter should have a cutoff frequency of about 4.5 kHz.

*Frequency modulation (FM)* is easily realized by PLLs and related circuits, too. A VCO is a frequency modulator by itself. Moreover, an ordinary linear or digital PLL is able to demodulate FM signals without additional circuitry. The demodulated signal can be taken from the output of the loop filter ( $u_f$ ; cf. Fig. 2.1 for example).

FM modulators and demodulators are easily converted to PM circuits with only a few additional components. As we already know, a VCO acts as an FM modulator. The instantaneous radian frequency  $\omega_2$  is

$$\omega_2(t) = \omega_0 + K_0 u_f(t)$$

hence is proportional to the information signal  $u_f$ . The phase of the VCO output is the integral of  $\omega_2$  over time

$$\varphi_2(t) = \omega_0 t + K_0 \int u_f(t) dt$$

which can be written as

$$\varphi_2(t) = \omega_0 t + \theta_2(t)$$

where  $\theta_2(t)$  represents the phase that carries the information. (The relationship between phase and radian frequency was explained in Sec. 2.2.) In PM,  $\theta_2(t)$  is required to be proportional to the information signal  $u_f$  *itself* and *not to its integral*. PM is therefore realized by inserting a differentiator between the information signal source and the control input ( $u_f$ ) of the VCO. In the same way, a PLL used as an FM receiver is converted to a PM decoder by integrating the  $u_f$  signal. The integrated  $u_f$  signal then represents the information signal. We should be aware, however, that an integrator is driven into saturation if an offset voltage exists at its input (which is the normal situation). Saturation is avoided when an “AC integrator” is used in place of an ordinary integrator, say, a circuit that integrates higher frequencies only but has finite gain at DC. The transfer function  $F(s)$  of an AC integrator can be given by

$$F(s) = \frac{1}{1 + sT_i}$$

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

where  $T_i$  is the integrator time constant. Roughly speaking, such a circuit would integrate signals whose radian frequency is above  $1/T_i$ , but would have gain 1 for lower frequencies. (Note: Such an “AC integrator” is nothing more than a simple first-order lowpass filter, which can be realized as a passive or active RC network.) It is quite clear that the bandwidth of such data transmissions cannot extend down to DC, but will be restricted to some lower band edge (maybe 300 Hz), as is the case with voice signals.

As mentioned, digital communications are of much greater interest today, so we will concentrate on digital applications.

The baseband transmission of digital signals has already been discussed in [Sec. 8.1](#), where we also considered various coding schemes such as NRZ, RZ, biphase, and delay modulation formats. We also became aware of the problem of recovering the clock frequency from the information signal and considered a number of circuits that extract clock information from the data. In broadband communications, bandpass modulation is applied almost exclusively, and we will now consider the techniques of bandpass communication in more detail.

## Amplitude Shift Keying

*Amplitude shift keying (ASK)* was one of the earliest methods of digital modulation used in radio telegraphy around the year 1900. In its simplest form, a high-frequency carrier is turned on and off by a binary data signal, which is also called “on-off keying.” (In the first era of communication technology, the transmission of Morse symbols effectively used a pseudo-ternary code: a “dot” was represented by a short carrier burst, a “dash” by a long carrier burst, and the pauses by “nothing.”) On-off keying is very simple, but has severe drawbacks. Switching the carrier on and off creates steep transients, which broadens the spectrum of the signal in an undesired way. Though quite primitive, ASK still finds applications where low bit rates are sufficient and low bandwidth is not of primary concern. In a typical ASK setup, a high-frequency carrier is switched by an NRZ encoded binary data stream. The receiver is built from an ordinary linear or digital PLL whose center frequency is tuned to the carrier frequency. The PLL is equipped with an in-lock detector (as shown in [Fig. 4.7](#) for example). Whenever the carrier is on, the PLL locks, and the output of the in-lock detector switches into the 1 state. When the carrier is off, the PLL unlocks, and the output of the in-lock detector goes to 0. Such receiver circuits are commonly referred to as *tone decoders*. A number of tone decoder ICs are available, such as the XR2213 (Exar) or the LM567 (National), and so on.

## Phase Shift Keying

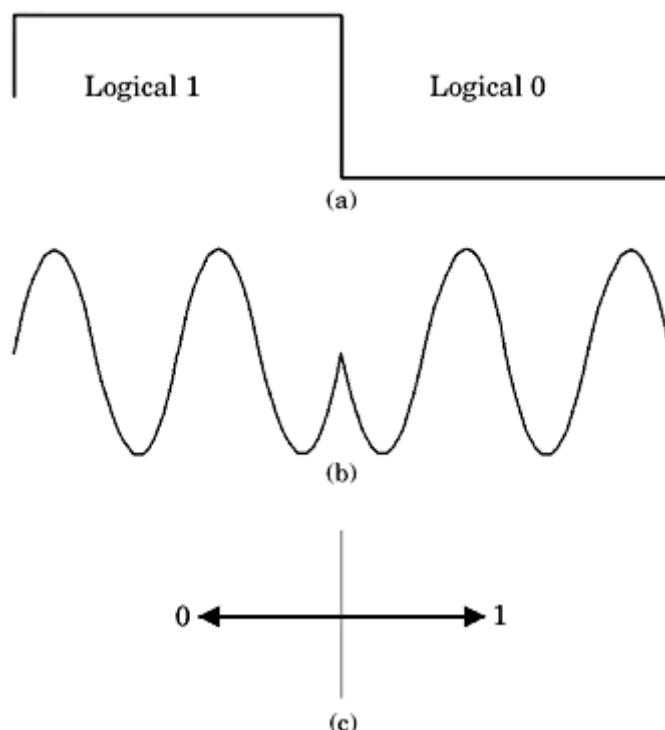
### Binary phase shift keying (BPSK)

Binary phase shift keying (BPSK) is the simplest type of digital phase modulation and is widely used in modern communication systems. Some of its descendants (QPSK, OQPSK, QAM) are more bandwidth efficient and will

**Any use is subject to the Terms of Use as given at the website.**

be discussed in the following sections. In classical PSK (also referred to as BPSK [binary PSK] or PSK<sub>2</sub>), the polarity of a carrier is controlled by a binary data signal. Such a binary signal is shown in Fig. 14.1a, and the modulated carrier in Fig. 14.1b. When the data signal is a logical 1, the carrier phase is 0 by definition. For a data signal of 0, the carrier phase becomes  $\pi$  ( $180^\circ$ ). The definition can be inverted if desired. As usual in the theory of alternating currents, the amplitude and phase of the modulated carrier can be represented as a vector (phasor) in the complex plane (see Fig. 14.1c). The length of the phasor represents its amplitude, and the angle with the horizontal axis its phase. For BPSK, the phase can only take two values 0 and  $\pi$ .

As we will see in Sec. 14.7, there is a distinct relationship between the symbol rate  $R$  (the number of binary symbols transmitted in one second) and the bandwidth  $W$  (in hertz) required to transmit the modulated signal. It will show up that for BPSK, the (two-sided) bandwidth  $W$  becomes approximately equal to  $R$ . Digital data are often carried by phone lines. Old analog telephone cables are known for their poor bandwidth, so if we planned to increase the symbol rate of an existing link by (say) a factor of 10, we cannot hope to reach that goal simply by increasing the carrier frequency by a factor of 10 and modulating it with the faster data signal. Therefore, we must try to increase the channel capacity (the throughput of binary symbols) without increasing the channel bandwidth. This seems contradictory at the first glance, but we will see shortly that it is possible.



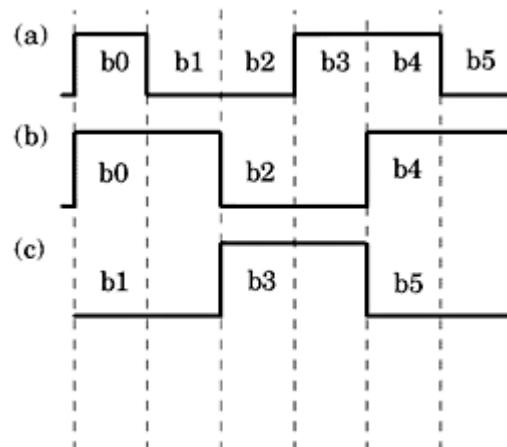
**Figure 14.1** Binary phase shift keying (BPSK). (a) Binary signal. (b) Modulated carrier. (c) Vector plot of the modulated carrier signal.

## Quadrature phase shift keying

QPSK (*quadrature phase shift keying*, also called *quaternary PSK* or  $\text{PSK}_4$ ) brings us a step further. How does it work? In BPSK, we created just one carrier and switched its polarity by a binary data stream. In QPSK, we generate *two* carriers that are offset in phase by  $90^\circ$ . One of the carriers is called the *in-phase* component; it is assigned to a phase of  $0^\circ$  and is mathematically represented by a *cosine wave*. The other carrier is called the *quadrature carrier* and is assigned to a phase of  $90^\circ$ . The latter is mathematically represented by a *sine wave*. (The ensemble of both carriers is often called a *complex signal*. This sounds a little bit curious, because a signal never can take on complex values—try to think of a complex temperature or humidity! The term complex only makes sense if we consider the in-phase component as the real part of a complex signal, and the quadrature component as the imaginary part thereof.)

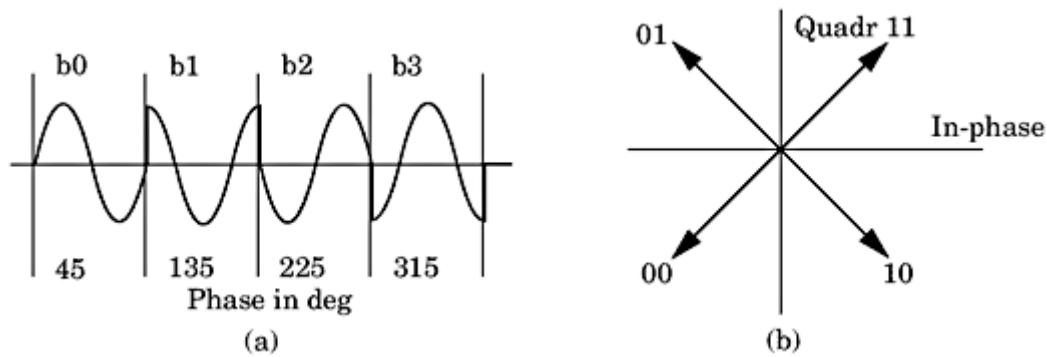
We are now going to switch the polarity of the in-phase carrier by one binary signal,  $s_1$ . Moreover we switch the polarity of the quadrature carrier by *another* binary signal,  $s_2$ . Hence, the phase of the in-phase carrier can take on the values  $0^\circ$  or  $\pi$  ( $180^\circ$ ), but the phase of the quadrature carrier can take on the values  $\pi/2$  ( $90^\circ$ ) or  $3\pi/2$  ( $270^\circ$ ). If we add the two carriers, the phase of the resulting signal can have values  $45^\circ$ ,  $135^\circ$ ,  $225^\circ$ , or  $315^\circ$ . Adding the two modulated carriers increases the symbol rate by a factor of two, but does not require additional bandwidth. Because the two carriers are “orthogonal” (perpendicular on each other), the two data signals  $s_1$  and  $s_2$  can be decoded by synchronous demodulation at the receiver, and therefore they do not interfere. The symbol rate of the QPSK signal is now half the symbol rate of the original data stream, as is easily seen from [Fig. 14.2](#). We note that with QPSK, 2 bits are transmitted in one symbol period.

[Figure 14.2](#) explains the principle of QPSK. The binary data stream as delivered by the data source is shown in [Fig. 14.2a](#). This sequence is partitioned into two data streams now: the even bits ( $b_0$ ,  $b_2$ , and so on) form the binary sequence that modulates the in-phase carrier ([Fig. 14.2b](#)), while the odd bits ( $b_1$ ,  $b_3$ , and so on) modulate the quadrature carrier ([Fig. 14.2c](#)). [Figure 14.3a](#) represents the four possible phases of the sum of in-phase and quadrature signals, while [Fig. 14.3c](#) shows the corresponding phasor plot.



**Figure 14.2** The principle of QPSK. (a) The binary signal to be transmitted. (b) The data

stream corresponding to the even bits in (a). (c) The data stream corresponding to the odd bits in (a).



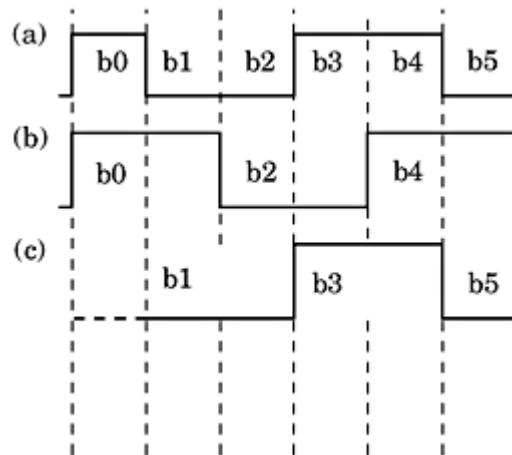
**Figure 14.3** The phase relationship of QPSK signals. (a) The modulated carrier can take on four different phases. (b) The phasor diagram of modulated carrier.

As can be seen from Fig. 14.2, both in-phase and quadrature data streams change their values at half the original symbol rate—in other words, on every second symbol delivered by the data source. Thus, the two data streams are *aligned*.

As we will see in Sec. 14.7, the bit streams (cf. Fig. 14.2b and c) are mostly lowpass filtered before modulating the in-phase and quadrature carriers. This is done in order to reduce the bandwidth of the modulated signal. Because the transitions of the bit streams are now smoothed, they vanish for a short interval of time when their amplitude changes from  $-1$  to  $1$ . Because both data streams are aligned, both modulated carriers can temporarily fade away. When a coherent demodulator (such as a *Costas loop*, for example) is used in the receiver to detect the data streams, this circuit could get out of lock. To avoid this drawback, OQPSK (*offset quadrature phase shift keying*) has been introduced. This will be treated in the next section.

## Offset quadrature PSK (OQPSK)

The bit signals plotted in Fig. 14.2b and c are modulating an in-phase and a quadrature carrier as described in Sec. 14.3.2. In OQPSK, the original bit sequence (Fig. 14.4a) is also partitioned into even (Fig. 14.4b) and odd (Fig. 14.4c) bits,



**Figure 14.4** The principle of OQPSK. (a) The binary signal to be transmitted. (b) The data stream corresponding to the even bits in (a). (c) The data stream corresponding to

the odd bits in (a). This data stream is shifted right by one bit cell compared with ordinary QPSK.

but the latter are delayed by one bit interval. Hence, not both data streams can change polarity at the same time, and the carrier cannot fade away at the bit transitions.

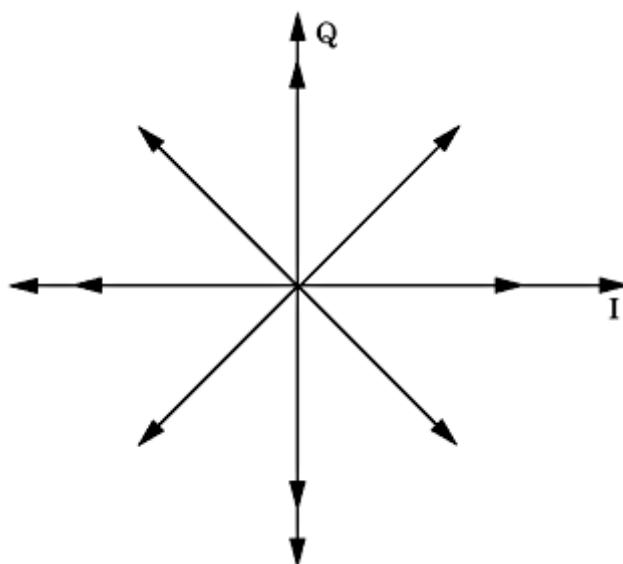
### m-ary PSK

The phase of the modulated carrier could have two or four different values in the modulation schemes considered hitherto. This technique can be expanded to an arbitrary large number of phase states.

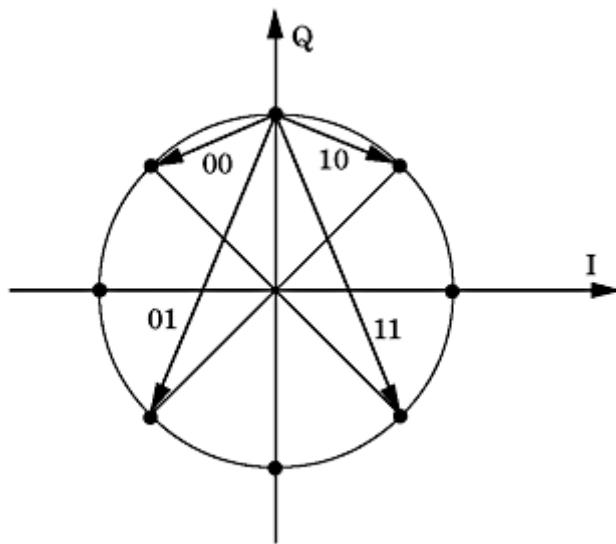
An example of 8-ary PSK is presented in Fig. 14.5. The phase of the modulated signal can be in one of eight different states; hence, three bits can be transmitted in one single symbol. Theoretically, the number of phase states could be increased arbitrarily, but a large number is chosen as the end points of the phasors come closer together. When noise is superimposed to the transmitted signal, the probability that the receiver takes a wrong decision also becomes larger. The increased bit error rate could be compensated—at least theoretically—by increasing the signal level. For every modulation scheme, there is a trade-off between symbol rate (the number of symbols transmitted per second) and signal power.

### Differential PSK (DPSK)

DPSK is an interesting variant of PSK. The PSK schemes considered hitherto used an absolutely encoded phase modulation scheme. When a 1 had to be transmitted in BPSK, for example, the phase was set to  $\pi$ ; otherwise, 0. When the next bit was another one, the phase of the carrier remained  $\pi$ , and so on. In differential BPSK, however, the phase of the carrier is *changed by  $\pi$*  when a 1 is transmitted, or left unchanged when a 0 is sent. When a series of 1s is transmitted, the phase changes by  $\pi$  in every bit cell. Differential BPSK (also called *differentially encoded BPSK*) has some benefit over absolutely encoded BPSK:



**Figure 14.5** A phasor diagram of 8-ary PSK.



**Figure 14.6** The phasor diagram of  $\pi/4$  DQPSK.

to demodulate the phase modulated signal in the receiver it is not required to know the absolute phase of the carrier, but the signal can be demodulated noncoherently, simply by comparing the phase in the current bit cell with the phase in the previous one. The demodulator can thus be realized with simpler hardware.

A special case of DPSK has gained attention—it was implemented in the North America Digital Cellular (IS-54) and in the Personal Digital Cellular (PDC) in Japan: the so-called  $\pi/4$  DQPSK scheme. This is a quadrature PSK code, similar to the QPSK discussed in [Sec. 14.3.2](#). With  $\pi/4$  DQPSK, two bits are encoded in every bit cell. The phase is *changed* by  $\pi/4$  when a bit pair 00 is transmitted, or by  $3\pi/4$  when the pattern 01 is transmitted, or by  $-\pi/4$  when 10 is transmitted, or finally by  $-3\pi/4$  when 11 is transmitted. This leads to the phasor diagram of [Fig. 14.6](#).

It is assumed that the initial phase of the carrier is  $90^\circ$  ( $\pi/2$ ). When the next bit pair to be sent is 00, the phase is incremented now by  $\pi/4$  or  $45^\circ$ . If the next bit pair had been 10, the phase would have been decremented by  $\pi/4$ , and so on. We recognize that eight different phases are possible with the  $\pi/4$  DQPSK code. This is the reason why this coding scheme is sometimes confused with 8-ary PSK as shown in [Fig. 14.5](#). With 8-ary PSK, however, 3 bits are transmitted in each bit cell, but in  $\pi/4$  DQPSK two bits are transmitted only. The  $\pi/4$  DQPSK can also be demodulated noncoherently as was the case with the simpler DBPSK.

## Frequency Shift Keying

### Binary FSK

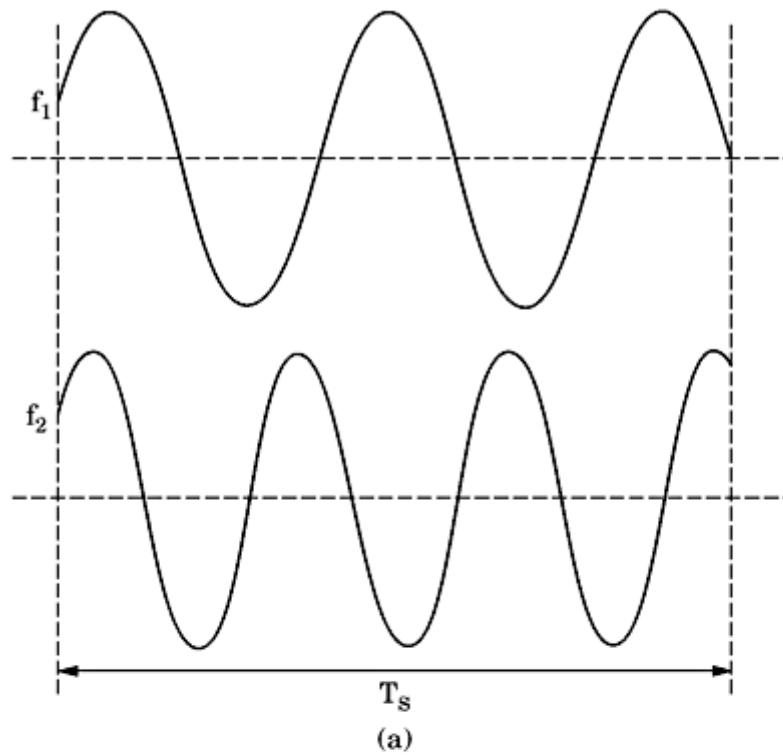
FSK (frequency shift keying) is another modulation scheme that is frequently used in modems. An FSK signal can be transmitted in the baseband, or it can be used to modulate a carrier. Let's discuss baseband communication first. In its simplest form,  $(\text{FSK}_2)$ , two different frequencies  $f_1$  and  $f_2$  are defined, where

---

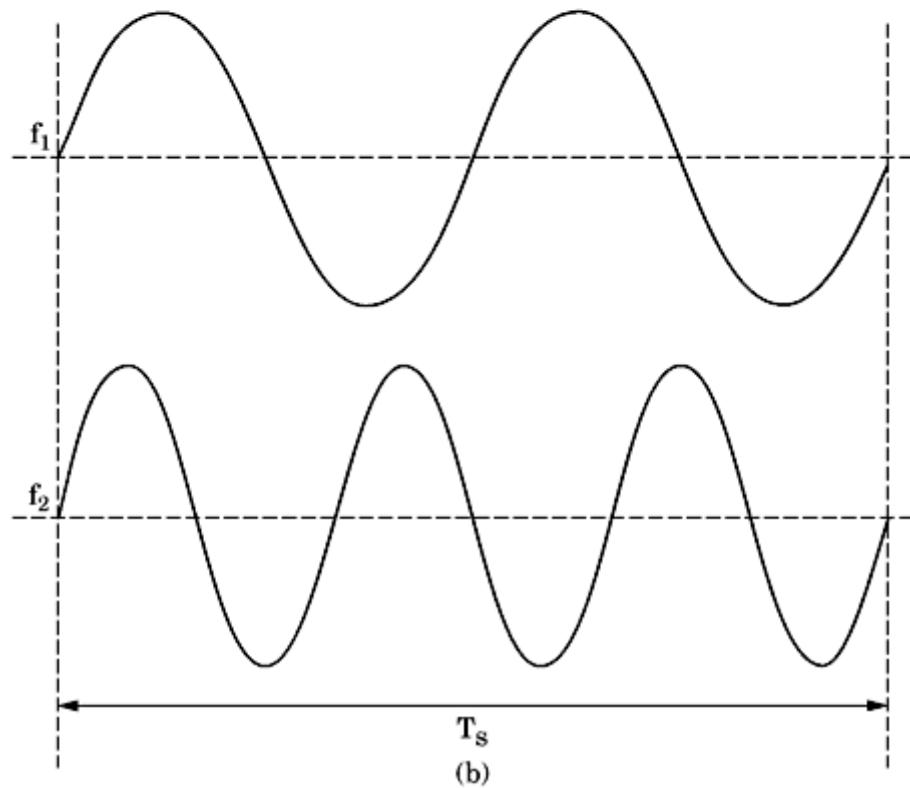
Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

---

$f_1$  represents a binary 0, and  $f_2$  represents a binary 1 or vice versa (see Fig. 14.7a). The two frequencies can be chosen arbitrarily—for example,  $f_1 = 1650$  Hz, and  $f_2 = 1850$  Hz. Moreover, it is not required that the symbol interval  $T_S$  is an integer multiple of one period of the  $f_1$  or of the  $f_2$  signal. Normally, a linear or digital



(a)



(b)

**Figure 14.7** The definition of binary FSK: (a) Nonorthogonal FSK: arbitrary values for  $f_1$  and  $f_2$  are chosen. (b) Orthogonal FSK: there is a tight relationship between frequencies  $f_1$  and  $f_2$  [cf. text].

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

PLL serves as an FSK decoder. The difference between the two frequencies must simply be chosen such that the PLL can safely determine whether it “sees” frequency  $f_1$  or  $f_2$ . This type of FSK is called nonorthogonal FSK. The other choice is *orthogonal FSK* (see Fig. 14.7b). Here, the frequencies  $f_1$  and  $f_2$  are chosen so as to fulfill the relationship

$$f_i = k_i \cdot \frac{1}{2T_S} \quad (14.1)$$

where  $i = 1, 2$  and  $k_i$  is a positive integer greater than 0. Both frequencies then are integer multiples of half the symbol rate  $1/(2T_S)$ . In the example of Fig. 14.7b,  $k_1 = 4$  and  $k_2 = 6$ . If we denote the signal waveforms for logical 0 and logical 1 by  $s_1(t)$  and  $s_2(t)$ , respectively, we have

$$\int_0^{T_S} s_i(t) s_j(t) dt = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases} \quad (14.2)$$

which means that the signals  $s_1(t)$  and  $s_2(t)$  are *orthogonal*. This property can be beneficial when the received symbols are buried in noise. A coherent FSK decoder then would store replicas of the symbol waveforms  $s_1(t)$  and  $s_2(t)$  and correlate the incoming symbol with both of these. Without noise, the correlation with one of the replicas would yield 1, and the correlation with the other 0. Whenever the correlation with  $s_1(t)$  leads to the result 0, the receiver would decide for logical 1. With added noise, the correlation coefficients depart from their ideal values 0 or 1, hence the receiver would decide for 0 if the correlation with  $s_1(t)$  becomes greater than the correlation with  $s_2(t)$  or for 1 if the reverse is true. Such a decision is a *maximum likelihood decision* in the statistical sense.

As will be discussed in Sec. 14.10, FSK can be demodulated coherently or noncoherently. Sklar has demonstrated<sup>20</sup> that when coherent detection is chosen, the minimum distance between the two frequencies  $f_1$  and  $f_2$  must be half the symbol rate—in other words,  $|f_1 - f_2|_{\min} = \frac{1}{2T_S}$ . When the signal is to be detected noncoherently, however, the minimum distance must be twice as large—meaning,  $|f_1 - f_2|_{\min} = \frac{1}{T_S}$ . In modern communication systems, orthogonal signaling is the preferred method of FSK.

## m-ary FSK

We only considered binary FSK hitherto, but FSK can be extended to m-ary FSK when more than two frequencies are chosen. For example, if  $m = 4$ , four different frequencies  $f_1$  to  $f_4$  would be used. When doing so, two bits per symbol period can be transmitted. There are a number of simple modems that use FSK<sub>2</sub> or FSK<sub>4</sub> and are working immediately with baseband signals. Of course, the baseband signal can be modulated onto a high-frequency

carrier. When doing so, many

FSK modulated carriers (with different carrier frequencies, of course) can share one single link.

As was the case in binary FSK, m-ary FSK can also be detected coherently or noncoherently. When coherent demodulation is chosen, the minimum distance between the signal frequencies must be half the symbol rate; for noncoherent demodulation, the minimum distance must equal the symbol rate.

### Minimum shift keying (MSK) and Gaussian MSK (GMSK)

*Minimum shift keying (MSK)* is a special case of binary FSK. First of all, it uses the minimum spacing between the two signal frequencies  $f_1$  and  $f_2$ —in other words, they are separated by half the symbol frequency  $1/(2 T_s)$ . Moreover, MSK uses *continuous-phase frequency shift keying (CPFSK)* which means that the phase of the modulated carrier is continuous at the symbol boundaries. Abrupt phase changes (as shown in Fig. 14.3a) are therefore avoided. For the following explications, we assume that a carrier having frequency  $f_0$  is used, and that a binary signal is transmitted at the symbol rate  $f_s = 1/T_s$ . When a logical 1 is transmitted, the frequency of the carrier is increased by  $f_s/4$ , and when a logical 0 is sent, the carrier frequency is decreased by  $f_s/4$ . The modulated carrier  $s(t)$  can then be expressed by

$$s(t) = \cos\left[2\pi\left(f_0 + \frac{d_k}{4T_s}\right)t + x_k\right], \quad kT_s < t < (k+1)T_s \quad (14.3)$$

Herein  $k$  is an index to the  $k$ th data bit, and  $d_k$  is  $+1$  when the  $k$ th bit is a logical 1 or  $-1$  when the  $k$ th bit is a logical 0. Consequently, when  $d_k$  is  $+1$ , the frequency of the signal  $s(t)$  gets  $f_0 + f_s/4$ , and for  $d_k = -1$ , the frequency of  $s(t)$  gets  $f_0 - f_s/4$ .  $x_k$  is a phase constant that is valid over the  $k$ th binary data interval. This constant is required to meet the continuous phase requirement. It is easily seen that  $x_k$  can be either  $0$  or  $\pi$ .  $x_k$  is computed by the recursive formula

$$x_k = \left[x_{k-1} + \frac{\pi k}{2}(d_{k-1} - d_k)\right] \bmod 2\pi \quad (14.4)$$

Applying the addition theorem for trigonometric functions to Eq. (14.3) leads to the alternative representation

$$s(t) = a_k \cos \frac{2\pi}{4T_s} t \cos 2\pi f_0 t - b_k \sin \frac{2\pi}{4T_s} t \sin 2\pi f_0 t \quad (14.5)$$

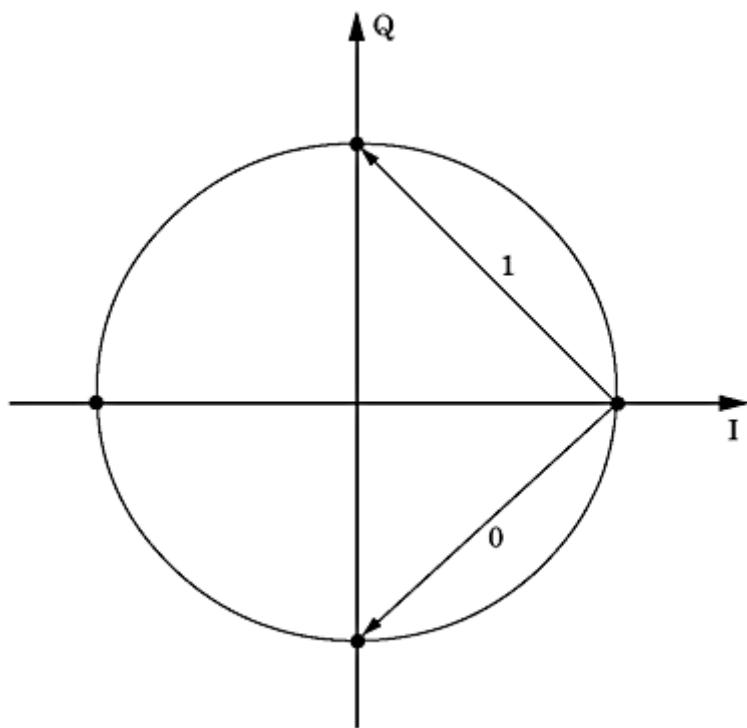
where

$$\begin{aligned} a_k &= \cos x_k = \pm 1 \\ b_k &= d_k \cos x_k = \pm 1 \end{aligned} \quad (14.6)$$

[Equation \(14.5\)](#) tells us that MSK can also be interpreted as *offset quadrature phase modulation*: the data dependent signal  $I_m(t) = a_k \cos \frac{2\pi}{4T_s} t$  modulates the in-phase carrier  $I_c(t) = \cos 2\pi f_0 t$ , and the data-dependent signal

$Q_m(t) = -b_k \sin \frac{2\pi}{4T_s} t$  modulates the quadrature carrier  $Q_c(t) = \sin 2\pi f_0 t$ . If the modulating signal  $I_m$  were made up simply by the constant  $a_k$ , and the modulating signal  $Q_m$  only by the constant  $b_k$ , the resulting modulation would be simply a QPSK scheme. But because the component  $I_m(t)$  modulating the in-phase carrier is identified as  $a_k \cos \frac{2\pi}{4T_s} t$ , the term  $\cos \frac{2\pi}{4T_s} t$  can be regarded as a *cosinusoidal symbol weighting* term. In analogy, the term  $\sin \frac{2\pi}{4T_s} t$  contained in the component  $Q_m(t)$  that modulates the quadrature carrier can be considered to be a *sinusoidal symbol weighting* term. It might appear that the data-dependent  $a_k$  and  $b_k$  terms can change every  $T_s$  seconds, since the source data,  $d_k$ , can change every  $T_s$  seconds. However, because of the continuous phase constraint, the  $a_k$  term can only change value at the zero crossings of  $\sin \frac{2\pi}{4T_s} t$ , and the  $b_k$  term can only change value at the zero crossings of  $\cos \frac{2\pi}{4T_s} t$ . Thus, the symbol weighting in either the  $I$ - or  $Q$ -channel is a half-cycle sinusoidal pulse of duration  $2T_s$  with an alternating sign. Because the symbol weighting terms in the  $I$ - and  $Q$ -channel are offset by  $T_s$  seconds from each other, the modulated signal  $s(t)$ , as given in Eq. (14.5), is the result of an OQPSK modulating scheme.

Knowing that the frequency of  $s(t)$  is incremented by 1/4 of the symbol frequency in a particular bit cell when a 1 is transmitted, we can state that the phase of  $s(t)$  is increased by  $\pi/2$  at the end of the symbol period. In analogy, the phase is decreased by  $\pi/2$  in a symbol period when a 0 is transmitted. This leads to the phasor and state transition diagram plotted in Fig. 14.8. Assuming



**Figure 14.8** A phasor and state transition diagram for MSK.

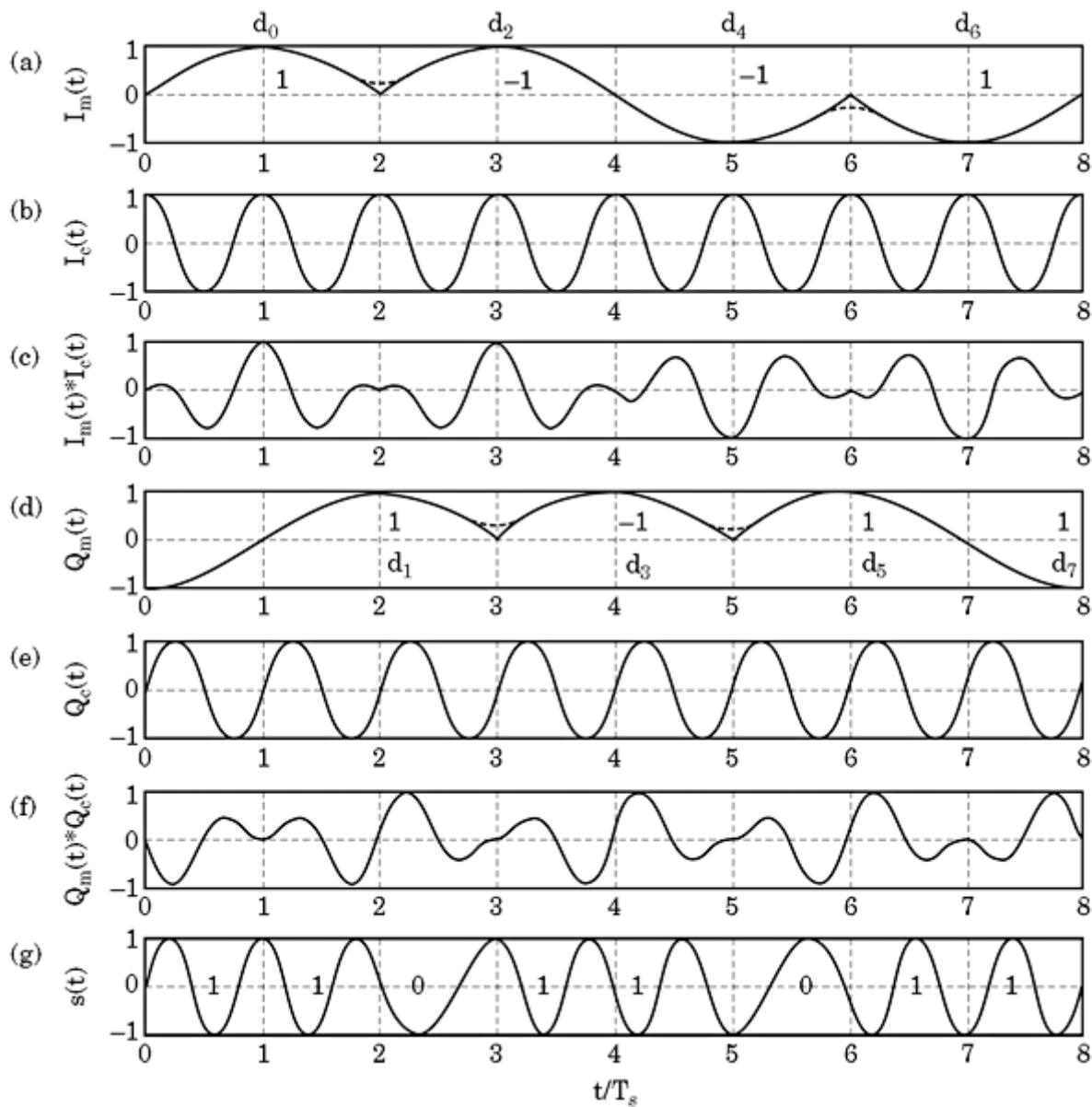
---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

that the carrier phase is 0 at the beginning of a data transmission, that phase can take four different values in the course of a transmitted bit stream—in other words, 0,  $\pi/2$ ,  $\pi$ , or  $3\pi/2$ . The state transitions are represented by the two arrows that symbolize a phase change of  $\pi/2$  or  $-\pi/2$ , respectively.

To illustrate the generation of an MSK signal, a numerical example is presented next. We assume that a binary data sequence 11011011 is transmitted with the symbol rate  $f_s = 1/T_s$ . To simplify the graphical representation, it is furthermore assumed that the carrier frequency  $f_0$  is equal to the symbol rate,  $f_0 = f_s$ . (In practice,  $f_0$  would be a multiple of  $f_s$ , of course. Think for example of a symbol rate of 200 kHz modulating a carrier of 2.4 GHz. This would describe a realistic scenario!)

Let's start with the modulating signal  $I_m$ , which is given by  $I_m(t) = a_k \cos \frac{2\pi}{4T_s} t$ . This is plotted in Fig. 14.9a. The polarity of the sinusoidal half-waves is



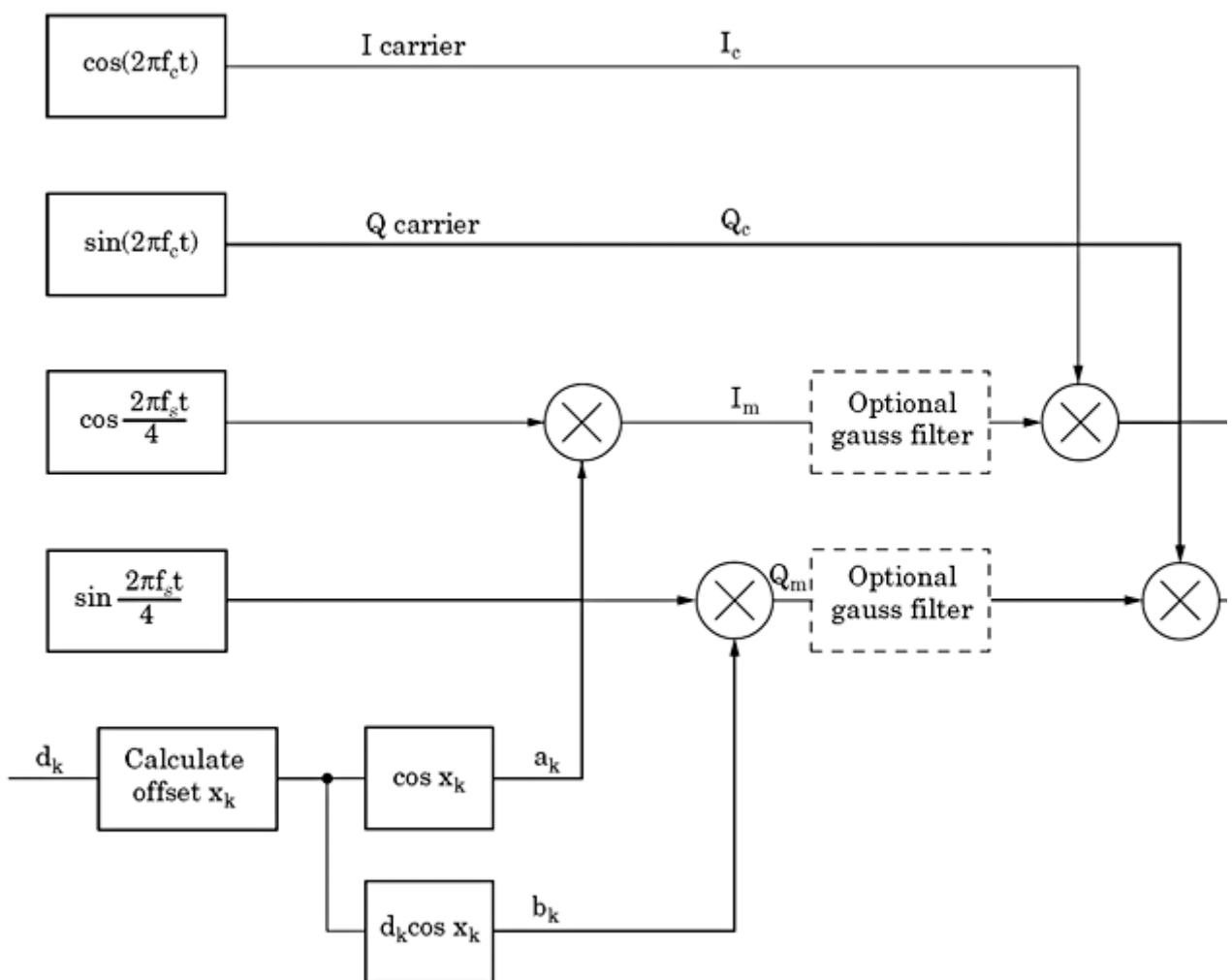
**Figure 14.9** Generation of an MSK signal  $s(t)$  signal by quadrature modulation. (a) The signal  $I_m(t)$  modulates the in-phase carrier. (b) The in-phase carrier  $I_c(t)$ . (c) The product of  $I_m(t) = I_c(t)$ . (d) The signal  $Q_m(t)$  modulates the quadrature carrier. (e) The quadrature carrier  $Q_c(t)$ . (f) The product  $Q_m(t) \cdot Q_c(t) =$  (g) The sum of signals in (c) and (f); this is the MSK signal  $s(t)$ .

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

determined by the  $a_k$  coefficients, which are determined from the value of  $x_k$  in the even data samples  $d_0, d_2, d_4 \dots$  [cf. Eq. 14.6]. Figure 14.9b shows the in-phase carrier  $I_c(t)$ . In Fig. 14.9c, the product  $I_m(t) \cdot I_c(t)$  is plotted. In Fig. 14.9d, the modulating signal  $Q_m$  is shown. The polarity of these half-waves is determined from the bits  $d_1, d_3, \dots$  and  $x_k$  in the odd samples [cf. Eq. (14.6)]. It modulates the quadrature carrier  $Q_c(t)$  [cf. Fig. 14.9e]. Figure 14.9f is the resulting product  $Q_m(t) \cdot Q_c(t)$ . Finally, the difference  $I_m(t) - Q_m(t) - Q_c(t)$  according to Eq. (14.5)—in other words, the signal  $s(t)$  is plotted in Fig. 14.9g. It is clearly seen that the MSK signal  $s(t)$  performs 5/4 cycles in those bit cells where  $d_k = 1$  and 3/4 cycles in the bit cells where  $d_k = -1$ . The value of the bit signal is indicated in every bit cell of Fig. 14.9g.

Figure 14.10 is one possible version of an MSK modulator. The two topmost blocks on the left produce the in-phase and quadrature carriers. The next two blocks generate the sinusoidal and cosinusoidal weighting signals; one half-wave of both of these have a duration of two symbols intervals ( $2T_s$ ). The data signal  $d_k$  is a bipolar signal whose value is +1 when a logical 1 is sent and -1 when a logical 0 is sent. The block labeled “Calculate offset  $x_k$ ” computes the offset constant  $x_k$  [cf. Eq. (14.4)].



**Figure 14.10** Generating an MSK signal using an offset quadrature PSK.

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

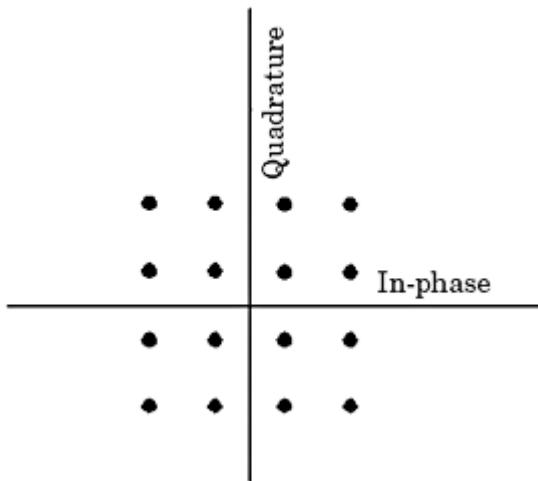
The blocks right from that block are computing the coefficients  $a_k$  and  $b_k$ , respectively [cf. Eq. (14.6)]. The bipolar signal  $a_k$  is then multiplied with  $\cos \frac{2\pi}{4T_s} t$  to create the  $I_m$  signal, and the bipolar signal  $b_k$  is multiplied with  $\sin \frac{2\pi}{4T_s} t$  to create the  $Q_m$  signal.  $I_m$  is multiplied with the I carrier, and  $Q_m$  is multiplied with the Q carrier. The difference of both modulated carriers according to Eq. (14.5) finally yields the output signal  $s(t)$ .

In normal MSK, the blocks in dashed rectangles are bypassed. Although the phase of the MSK signal  $s(t)$  is continuous at the symbol boundaries, the slope of the phase changes abruptly at these instants of time. This creates relatively large side lobes at higher frequencies in the spectrum of the MSK signal. When the modulating signals  $I_m$  and  $Q_m$  are lowpass filtered, however, these slope changes are smoothed, and the side lobes are drastically reduced.<sup>63</sup> To avoid overshoot, Gauss lowpass filters are used for this purpose. The impulse response of the Gauss lowpass filter is a Gaussian function of the form  $\exp(-t^2/a^2)$ , where  $a$  is a constant. The smoothing effect of the Gauss filter can be seen in Fig. 14.9a and d; here, the waveforms without Gauss filtering are drawn by solid lines, whereas the smoothed waveforms are plotted with a dotted line.

The 3-dB cutoff frequency of the Gauss filter is usually chosen  $f_{3\text{dB}} \approx 0.3 f_s$ —in other words, about 0.3 times the symbol frequency.

## Quadrature Amplitude Modulation (QAM)

Having increased the channel capacity by a factor of two without sacrificing bandwidth, we will now expand this principle to even greater benefit. The amplitudes of the two carriers for QPSK have always been the same, so we can normalize these amplitudes to 1. What about using more than one amplitude value though? Say 1 or 2? When doing so, the modulated in-phase carrier could take on a phase of 0 with an amplitude of 1, or a phase of 0 with an amplitude of 2, or a phase of  $180^\circ$  with an amplitude of 1, or a phase of  $180^\circ$  with an amplitude of 2. The same would hold true for the quadrature signal. If the receiver is not only able to determine the phases of the two signal components but also be in the position to discriminate different signal amplitudes, this would increase the channel throughput by another factor of 2. In other words, if we compare the channel capacity with ordinary BPSK, this modulation scheme would transmit four bits in one symbol period. This modulation scheme is called QAM (*quadrature amplitude modulation*). Actually, QAM is a combination of PSK with ASK, because we alter both phase and amplitude. Of course, we must not restrict the amplitudes to only two values—more values can be used. A phasor diagram of QAM can look like Fig. 14.11. Here, two amplitude values are used. The end points of the phasors have identical distance on the horizontal and on the vertical axis. Note that the in-phase component can take on relative amplitudes of 0.5 and 1.5 and not 1 and 2, respectively. With this choice of amplitudes, all phasors become equidistant on both axes. Using the scheme shown in Fig. 14.11, the combined carrier can take on 16 states, hence four bits



**Figure 14.11** A phasor diagram of QAM.

per symbol are transmitted. The code shown in Fig. 14.11 is commonly called QAM<sub>16</sub>. Note that QAM<sub>4</sub> is identical with QPSK. Theoretically, the set of amplitudes can be made as large as desired, but noise superimposed to the transmitted signal sets limits. To discriminate between different amplitude levels, the receiver must define *decision limits*. With reference to Fig. 14.11, the receiver would split the phasor plane into a number of squares of equal size, and the marked dots would sit in the center of each square. Noise on the signal would cause the phasor to deviate from its nominal position. As long as the end point stays within the corresponding decision region, no error would result. Errors occur, however, when the noise causes the phasor to migrate into another decision region. The actual bit error rate of a particular code can be computed using statistics and Shannon's information theory.<sup>20</sup> Today, QAM is the most widely used technique. In modern modems, QAM<sub>64</sub>, QAM<sub>128</sub>, and QAM<sub>256</sub> come into play. As we will see in Sec. 14.9, QAM will be used very efficiently in DVB (Digital Video Broadcasting).

## The Role of Synchronization in Digital Communications

Before discussing the various modulation techniques applied in digital communication, some notes on the synchronization problem appear appropriate. Whenever digital data are transmitted by bandpass modulation, synchronization on different signal levels will be required. Assume for the moment that binary phase shift keying (BPSK) is used to modulate the phase of a high-frequency carrier. At the receiver, the data signal must be recovered by demodulation. In most cases, the demodulation is performed synchronously—in other words, the receiver generates a replica of the carrier and multiplies the incoming signal with that of the reconstructed carrier. This shifts the spectrum of the received signal by a frequency offset which is equal to the carrier frequency, and the data signal is obtained by simple lowpass filtering. Because the replica of the carrier must be in phase with the latter, *phase synchronization* is required.

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

This can be considered the first level of synchronization and is usually realized by PLL circuits. For reasons to be explained in the next sections, locking onto a modulated carrier is not always trivial, so extended versions of PLLs are required in many cases (for example, the Costas loop).

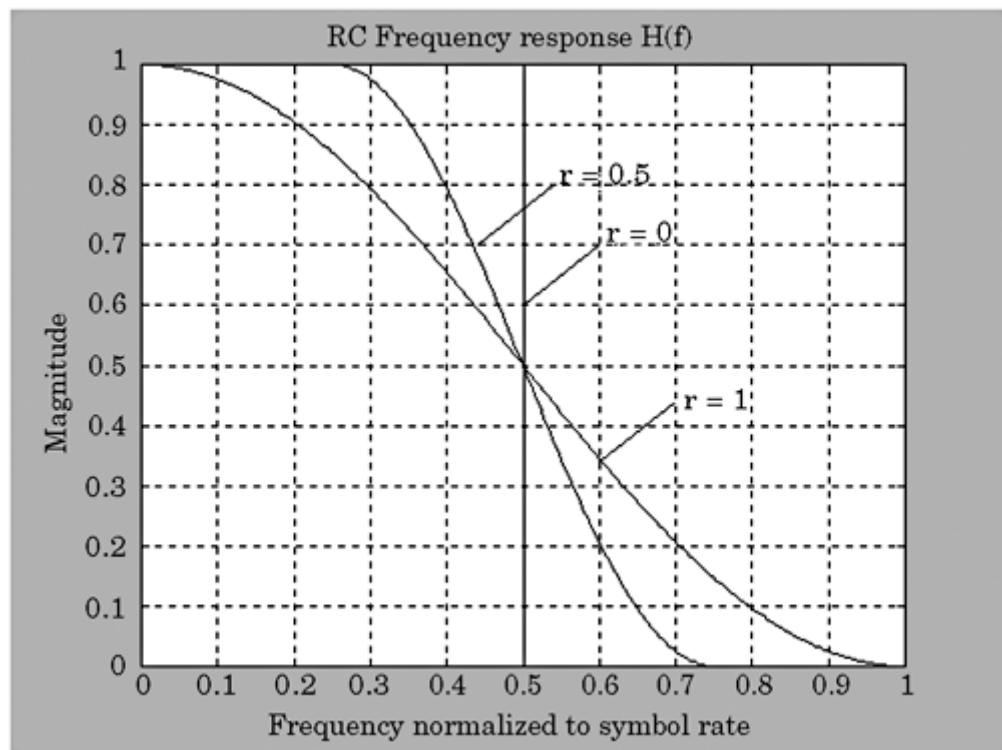
After demodulation, the data stream in the receiver is almost always a filtered version of the original data signal. While the original data signal represented a square-wave signal, the transients of the filtered data signal become smoothed. In many situations, the receiver performs a correlation of the received data with template functions that are stored in the receiver. This is done to reduce the effects of noise, which has been added to the signal on the transmission link. This correlation is usually performed during a time interval that must be identical with the symbol duration, hence the receiver must know when an incoming symbol starts and when it is over. A second level of synchronization must be performed then, which is referred to as *symbol synchronization*. The circuits that perform symbol synchronization are still other special versions of PLLs; we will discuss some of them in following sections, including the *Early-Late-Gate* synchronizer.

In some communication systems, an even higher level of synchronization is required. This is usually called *frame synchronization*. This kind of synchronization comes into play when the information is organized into blocks. Such a block may consist of a block header, followed by the actual data and eventually by a trailer that contains check code (for example, a cyclic redundancy check, CRC). Information is organized in blocks when the communication channel is time-shared by a number of transmitters. In this case, the receiver must be able to decide when a block starts. Frame synchronization will not be discussed here.<sup>20</sup>

## Digital Communications Using BPSK

### Transmitter considerations

**Filtering the data: yes or no?** Communicating binary signals by binary phase shift keying looks quite trivial, but imposes serious problems if we consider bandwidth requirements. As shown in Fig. 14.1, the polarity of a high-frequency carrier is switched by the binary information sequence. If the data are not filtered, the carrier is modulated by a square wave. It is not strictly a square wave, but rather a binary random sequence that can change its amplitude at the start of each symbol period. What the data signal has in common with a square wave are its very steep transients. Normally, the NRZ format is used to modulate the carrier (cf. Fig. 8.1). If we transmitted an infinite sequence of binary 0s or 1s, the binary random sequence would degenerate to a DC level, hence the bandwidth of the data signal would be zero. Because we actually send an arbitrary sequence of 0s and 1s, the spectrum of the data signal will contain some higher frequencies. It is easy to recognize that the frequency of the data sequence becomes maximum when we transmit a sequence of alternating 1s and 0s. In this case, the data signal would be a square-wave signal whose frequency is half the symbol rate.



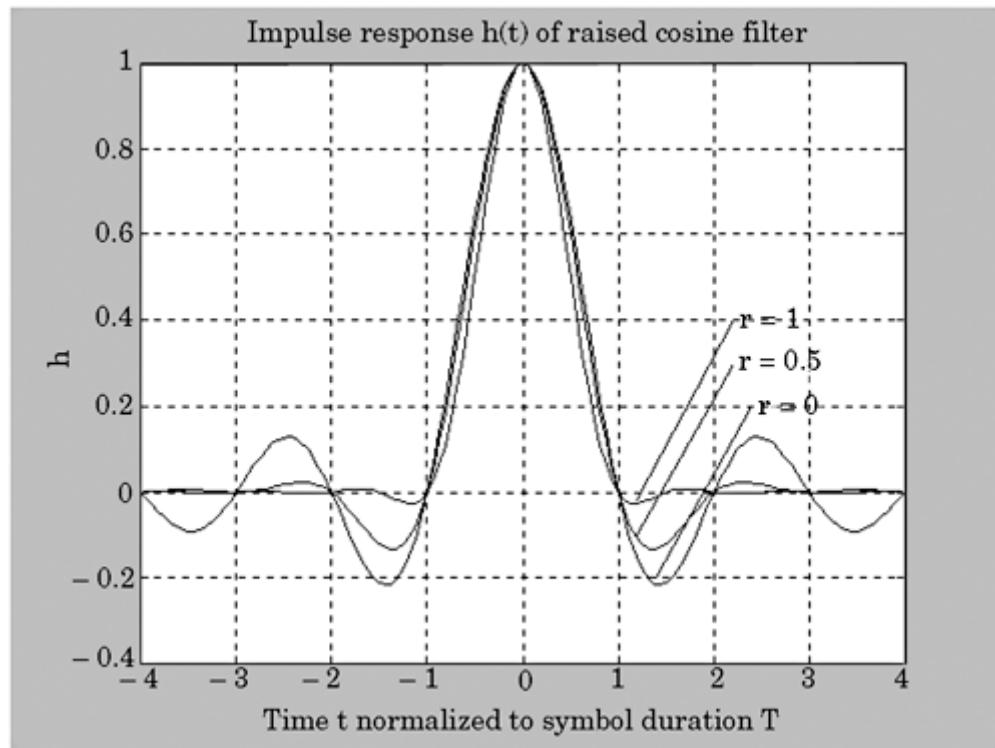
**Figure 14.12** The amplitude response of a brickwall filter ( $r = 0$ ) and raised cosine filter ( $r > 0$ ).

As Nyquist realized in 1928,<sup>40</sup> a receiver will be able to detect the data signal if only the fundamental component of this square-wave signal is transmitted. Consequently, we would pass the data signal through a lowpass filter at the transmitter site before modulating the carrier. The most appropriate filter would be a “brickwall filter,” (cf. Fig. 14.12) which is an ideal lowpass filter having gain 1 at frequencies below half the symbol rate and gain 0 elsewhere. The impulse response of the brickwall filter is

$$h(t) = \frac{\sin(\pi t/T)}{\pi t/T} \quad (14.17)$$

which is referred to as a *sinc* function.  $T$  is the symbol period. The duration of the impulse response is infinite, and it starts at  $t = -\infty$ ; the filter delay is also infinite. The impulse response is plotted in Fig. 14.13 by the curve labeled  $r = 0$  (the meaning of  $r$  will be explained in the following). Of course, such a filter cannot be realized. An approximation to the brickwall filter can be implemented by a FIR digital filter (FIR = finite impulse response [cf. App. C]) to any desired level of accuracy. However, because the impulse response decays slowly to 0, this leads to excessively long FIR filters—that is, to filters with a large number of taps. Nevertheless, we will consider the impulse response in more detail because it exhibits a very useful property.

**Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.**



**Figure 14.13** The impulse response of the raised cosine filter for various values of (relative) excess bandwidth  $r$ .

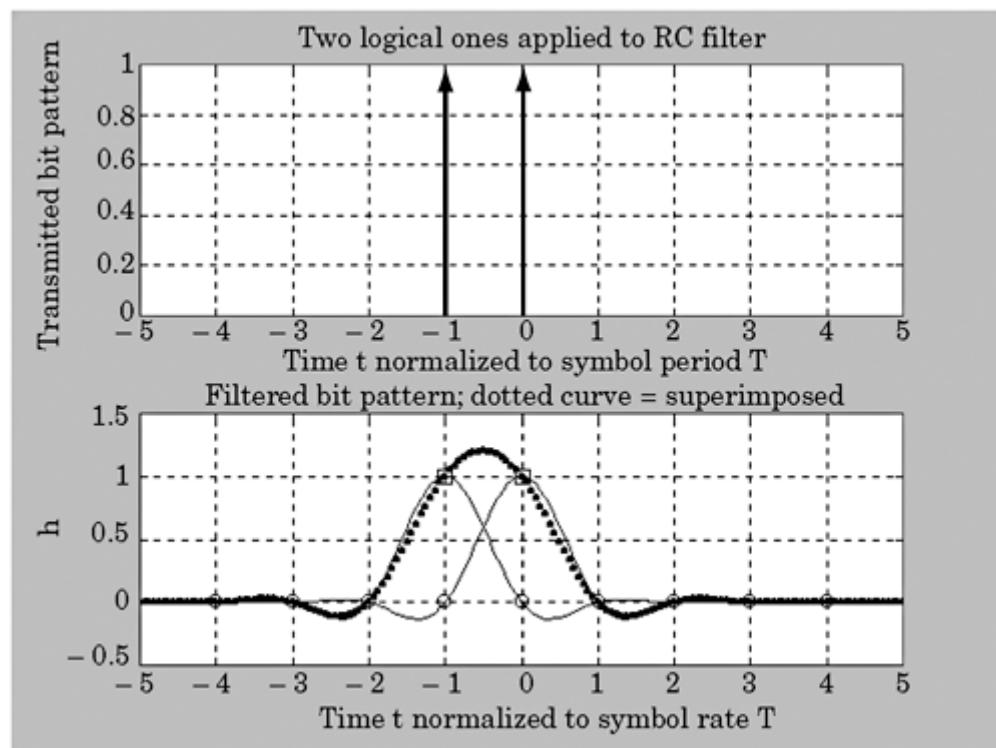
Every FIR filter causes the signal that is passed to be delayed, where the delay  $\tau$  is given by

$$\tau = \frac{L - 1}{2} T_F \quad (14.18)$$

$L$  is the length of the FIR filter, and  $T_F$  is the sampling interval of the filter. In Fig. 14.13, the delay of the filter has been neglected, hence its maximum ( $\pi/2$ ) occurs at time  $t = 0$ . Normally, such a filter would be sampled at a frequency  $f_F$ , which is an integer multiple of the symbol rate  $R = 1/T$ , where  $T$  is the symbol period. If we used a FIR filter of length 33, for example, it would have a delay of  $16 T_F$ , and if we assume, furthermore, that the filter sampling frequency  $f_F$  is four times the symbol rate  $R$ , the filter would delay the signal by four symbol periods. For this FIR filter, the impulse response  $h(t)$  would no longer be symmetrical about  $t = 0$ , but rather about  $t = 4T$ . For simplicity, the delay has been omitted in Fig. 14.13—in other words, we assume that the filter is a so-called *zero phase filter*, which causes no delay. Note that the impulse response of the brickwall filter becomes 0 at  $t = T, 2T, 3T$ , and so on. Understand, too, that the impulse response is nothing more than the response of the filter to a delta function with amplitude 1, applied at  $t = 0$  to its input. Sending a logical 1 therefore corresponds to applying a positive delta function, and sending a logical 0 corresponds to applying a negative delta function (amplitude  $-1$ ). Next, we consider the case where a number of symbols is passed through the filter in succession. If two succeeding 1s are supplied, the first

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

logical 1 will be represented by a delta function with amplitude +1 applied at  $t = -T$ , and the second logical 1 will be represented by another delta function with amplitude +1 applied at  $t = 0$ . This situation has been plotted in Fig. 14.14. The upper trace shows the two delta functions (marked by arrows), while the lower trace presents the corresponding impulse responses (solid curves). The zeroes of the impulse responses are marked by circles. The superposition of the two impulse responses is shown by the dotted curve (thick). In order to recover the data, the receiver must sample the filtered signal exactly at time  $t = -T, 0, T$ , and so forth. As can be seen from Fig. 14.14, the amplitude of the combined signal at  $t = -T$ , (marked by a square) depends uniquely on the impulse applied at  $t = -T$ ; the impulse response caused by the second delta function is zero at this time. The same holds for the combined signal at  $t = 0$ . Its amplitude at  $t = 0$  (marked by a square) uniquely depends on the amplitude of the second delta function and does not contain any contribution from the first delta function. In other words, the impulse responses caused by various delta functions do *not interfere*—in other words, when using the brickwall filter there will be no *inter-symbol interference* (ISI). To recover the data signal without error, the applied filter must be chosen such that no ISI can occur. (It is easy to demonstrate how ISI can be created: assume that we filter the data signal by a Butterworth lowpass filter, for example. Its impulse response will be a damped oscillation. It also passes



**Figure 14.14** The response of the raised cosine filter to a sequence of two binary ones applied to its input. Upper trace: a sequence of two delta functions corresponding to two binary 1s transmitted in succession. Lower trace: the solid curve (thin) whose maximum is at  $t/T = -1$  is the response of the filter to the delta function applied at  $t/T = -1$ , the solid curve whose maximum is at  $t/T = 0$  is the response to the delta function applied at  $t/T = 0$ , and the thick (dotted) curve is the superposition of these two responses.

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

through zero, but these zeroes are not at  $t = 0, T, 2T$ , and so on; hence, the impulse responses coming from delta functions applied to the input will interfere, or, to put it another way, an output sample taken at  $t = kT$  [ $k$  = positive integer] is the sum of the contributions of many symbols. This cannot be tolerated, of course.)

**How to get rid of ISI?** As we have seen, a useful approximation to a brickwall filter would lead to excessive filter length. Nyquist has shown a valuable alternative: if the transition region of the filter is widened, its impulse response decays faster toward zero. Moreover, if the amplitude response  $H(f)$  of the filter is symmetrical about half the symbol rate ( $R/2$ ), the locations of the zeroes of its impulse response remain unchanged. Filters having this property are commonly referred to as *Nyquist filters*. One possible realization of the Nyquist filter is the *raised cosine filter (RCF)*. Its transition region (the region between passband and stopband) is shaped by a “raised” cosine function—in other words, by a cosine wave that stands “on a pedestal.” The width of the transition band is determined by the parameter  $r$ , which is called excess bandwidth. If the frequency corresponding to half the symbol rate is denoted  $W_0$  ( $W_0 = 1/2T$ ), the transition band starts at  $f = (1 - r) W_0$  and ends at  $(1 + r) W_0$ . The amplitude response  $H(s)$  of the RCF has been plotted in [Fig. 14.12](#) for three values of  $r$ ,  $r = 0, 0.5$ , and  $1$ . (The case  $r = 0$  applies for the brickwall filter.) Note that the frequency response is symmetrical about half the symbol rate. Mathematically, the frequency response of the RCF is given by

$$H(f) = \begin{cases} 1 & \text{for } |f| < \frac{1-r}{2T} \\ \cos^2 \frac{\pi}{4} \frac{2T|f| + r - 1}{r} & \text{for } \frac{1-r}{2T} \leq |f| \leq \frac{1+r}{2T} \\ 0 & \text{for } |f| > \frac{1+r}{2T} \end{cases} \quad (14.19)$$

[Figure 14.13](#) shows the impulse response of the RCF for  $r = 0, 0.5$ , and  $1$ . The larger  $r$  is chosen, the faster the decay becomes. To economize bandwidth, however, large values of  $r$  must be avoided. In practice, values of  $r$  in the range of  $0.15$  to  $0.35$  are customary. For completeness, we also give the expression for the impulse response  $h(t)$  of the RCF:<sup>41</sup>

$$h(t) = \frac{\sin(\pi t/T) \cdot \cos(\pi rt/T)}{(\pi t/T) \cdot \left[ 1 - \left( \frac{2rt}{T} \right)^2 \right]} \quad (14.20)$$

Designing an FIR raised cosine filter is an easy task: [Eq. \(14.20\)](#) can be used to compute the filter coefficients. Care must be taken, however, since there exist values of  $t$  where both numerator and denominator become 0. This happens if the expression in the square brackets of the denominator becomes 0, explicitly for  $t/T = 1/(2r)$ . For the same value of  $t$ , the cosine term also becomes 0. This

**Any use is subject to the Terms of Use as given at the website.**

singularity is removed by replacing both numerator and denominator with their derivatives; mathematically, this is called L' Hôpital's rule. The computation is made even easier if Matlab's Signal Processing Toolbox is available.<sup>25</sup> It contains a function called FIRRCOS that calculates the coefficients of the RCF.

So far, the raised cosine filter seems to offer the optimum solution for lowpass filtering because it completely suppresses ISI. As we will see in Sec. 14.7.2, the receiver also needs a lowpass filter for different reasons. It will become evident that we cannot use another raised cosine filter at the receiver: when doing so, the data signal must pass through two cascaded RCFs—one in the transmitter and one in the receiver. The overall frequency response then would be the RCF transfer function *squared!* The resulting frequency response would no longer be symmetrical about half the symbol rate, and ISI would occur. We will see in the next section that the so-called *root raised cosine filter* (RRCF) will fix that problem.

## Receiver considerations

A number of tasks must be performed in the receiver to reconstruct the symbol stream. The most important of these will be discussed here. Because the data modulate the carrier, the symbol stream must be demodulated by a convenient procedure. Basically, we can differentiate between coherent (synchronous) and noncoherent demodulation. Coherent demodulation is more efficient, especially if noise has been added to the signal when it propagated across the data link. When coherent detection must be applied, the receiver must know the phase of the carrier exactly—meaning, the receiver must reconstruct a replica of the carrier. As we will see in the following, this is more difficult than it first appears. If noncoherent detection is to be realized, the receiver must not know the phase of the carrier. It rather takes the phase of the currently detected symbol as a reference for the detection of the next symbol. If the phase does not change, the receiver knows that the next symbol is the same as the previous one. Otherwise, it decides that the symbol changed its value from 0 to 1, or from 1 to 0. For correct operation, some initial synchronization becomes necessary—thus, the receiver must know at the start of a transmission whether the currently received symbol is a 0 or a 1. Noncoherent detection has been extensively analyzed in ref 20. Due to its higher noise immunity, coherent detection is the preferred method in most receiver circuits today,<sup>42</sup> so we will concentrate on this technique.

**Phase synchronization within the receiver.** When working with BPSK the carrier is modulated by a (filtered or unfiltered) symbol sequence. The carrier has the frequency  $f_s$ , and its spectrum consists of just one line at  $f_s$ , but the symbol sequence has a broader spectrum that ranges approximately from 0 to half the symbol rate when it is filtered. When it is not, the spectrum can be much broader. Assume for the moment that the data signal consists of only one frequency  $f_d$ . With BPSK, the data signal and carrier are multiplied, hence their spectrum will contain lines at the sum frequency  $f_s + f_d$  and at the difference frequency  $f_s - f_d$ .

Normally, the data signal is an arbitrary sequence of logical 1s and 0s, where a 1 is represented by a positive voltage (+1) and a 0 by a negative voltage (-1). On a long run, there will be approximately the same number of 1s and 0s, hence this signal will have no DC component or only a very small one. Consequently, the spectrum of the data signal has almost no power at  $f = 0$ . This implies that the modulated carrier does not contain appreciable power at the carrier frequency; the power is concentrated at other frequencies. This just means we cannot use a simple PLL to reconstruct a replica of the carrier!

Quite an arsenal of solutions has been invented to overcome that problem.<sup>1,20</sup>

**The squaring loop.** One of the earliest circuits was the *squaring loop* (see Fig. 14.15). The radian frequency of the carrier is  $\omega_1$  here, and the carrier is multiplied by the symbol signal  $m(t)$ . In the simplest case,  $m(t)$  is a binary random sequence taking on the values +1 or -1, respectively. The modulated carrier signal  $s(t)$  is therefore represented by

$$s(t) = m(t) \sin(\omega_1 t)$$

The output of the (analog) squaring device is then

$$s^2(t) = m^2(t) \sin^2(\omega_1 t)$$

Making use of the multiplication theorems of trigonometry, this consists of a DC term (which can be discarded here) and a high-frequency term of the form  $\cos(2\omega_1 t)$ , which is a term having twice the frequency of the carrier. A conventional

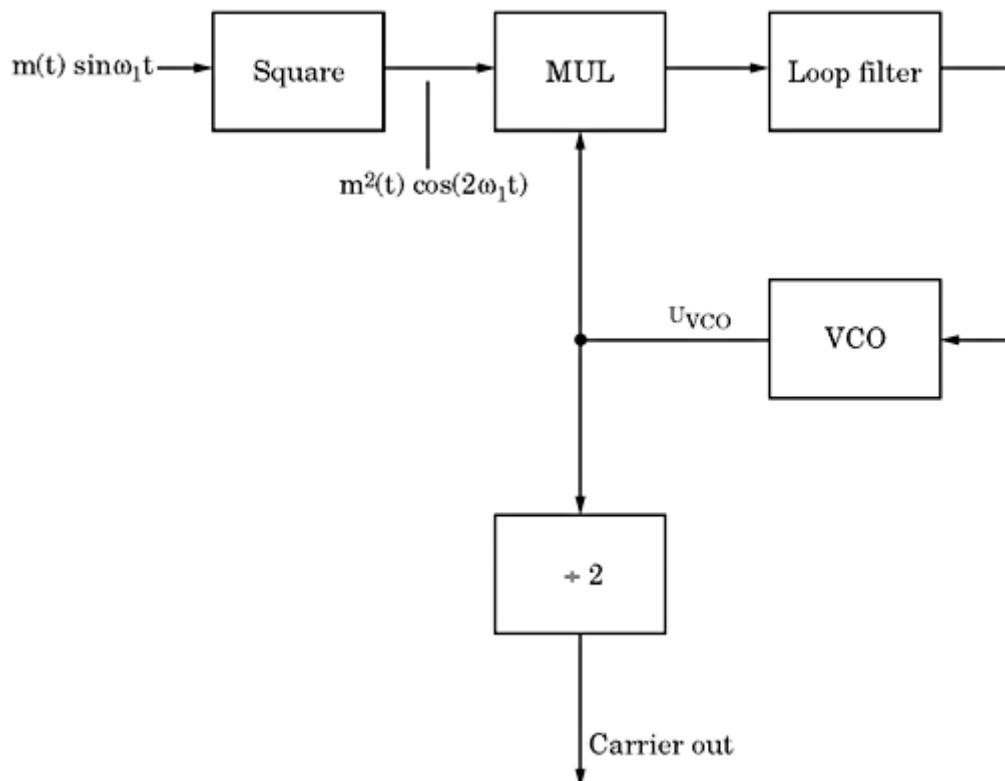


Figure 14.15 A squaring loop.

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

PLL is then used to lock onto that frequency. A replica of the carrier is obtained by scaling down that signal by a factor of 2.

**The Costas loop.** Because squaring devices were hard to implement using analog circuitry, alternatives have been searched for. A very successful solution is the *Costas loop*.<sup>1,20</sup> It originally was realized as an analog circuit, but is implemented mostly by digital techniques today. An example is the digital Costas loop HSP50210, originally manufactured by Harris.<sup>42</sup> The basic Costas loop is shown in Fig. 14.16. The explanation becomes easier if we assume that the circuit has locked. The input signal can be represented by  $m(t) \sin(\omega_1 t + \theta_1)$ , and the reconstructed carrier (the output signal of the VCO) by  $\sin(\omega_1 t + \theta_2)$ . In the locked state, the phases  $\theta_1$  and  $\theta_2$  will be nearly the same. Let's see how the Costas loop manages to lock onto the carrier frequency. Two closed loops are recognized, one in the upper branch (*I* branch) and one in the lower branch (*Q* branch). Assume for the moment that only the lower loop does exist, and discard the multiplier at center right—thus, we connect the output of the loop filter in the lower branch directly to the input of the center lowpass filter. This forms a rather conventional PLL. The multiplier at the left in the *Q* branch acts as a phase detector. This PLL circuit would adjust the frequency of the VCO in such a way that it is phase-locked to the carrier. The output of the VCO would have nearly the same phase as the input signal ( $\theta_1 \approx \theta_2$ ), and the two input signals of the lower multiplier would be out of phase by  $90^\circ$ , normal for linear PLLs. This arrangement would work as long as the modulating signal  $m(t)$  does not change polarity. When  $m(t)$  changes polarity, however, the output of the multiplier is inverted as well, which means that the multiplier no longer sees a phase error  $\theta_e = \theta_1 - \theta_2$ , but rather a phase error of  $\theta_e = \theta_1 - \theta_2 + \pi$ . Consequently, the loop would lock in antiphase with the carrier now, which is not the desired action. Rather, we are looking for a circuit that does not change polarity when  $m(t)$  is

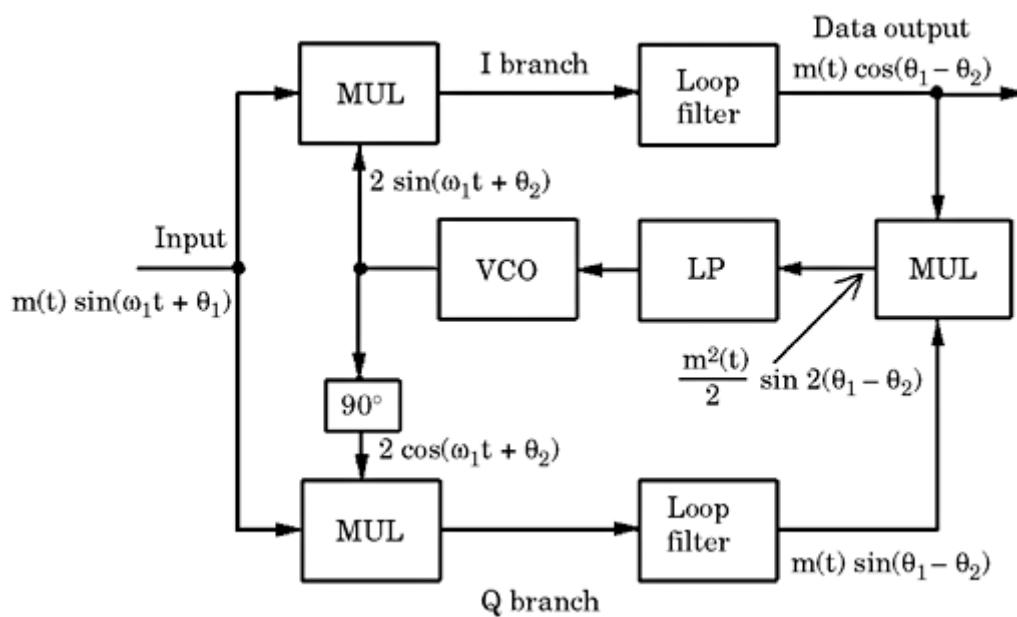


Figure 14.16 The Costas loop for BPSK.

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

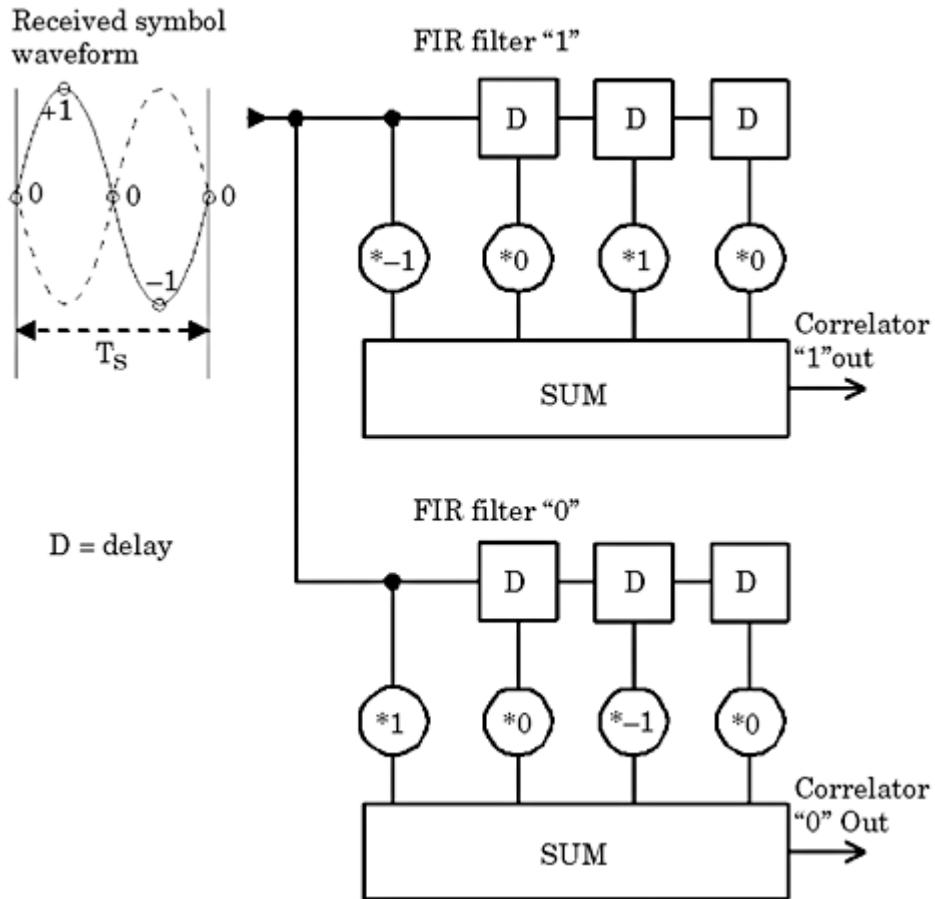
inverted. To overcome the problem, the  $I$  branch is added, together with the additional multiplier at center right. As indicated in Fig. 14.16, the output signal of the loop filter in the  $I$  branch is given by  $m(t) \cos(\theta_1 - \theta_2)$ . If phase is nearly correct for tracking (in other words,  $\theta_1 - \theta_2$  is small), the cosine term in the  $I$  branch is nearly 1, thus the output of the upper loop filter is the demodulated signal  $m(t)$ . Because the polarity of the output signal changes with  $m(t)$ , it is used to cancel the change in polarity at the output of the lower loop filter. Multiplying both loop filter output signals ( $I$  and  $Q$  branch) creates a signal of the form

$$\frac{m^2(t)}{2} \sin 2(\theta_1 - \theta_2)$$

hence, a signal that is proportional to the phase error  $\theta_e$  when the loop is locked. When lock is achieved, the output of the VCO is in phase with the carrier, so the multiplier in the  $I$  branch synchronously demodulates the data signal. The names of the branches are self-explanatory: the reconstructed carrier fed to the  $I$  branch is the in-phase component, and the reconstructed carrier fed to the  $Q$  branch is the quadrature component.

**Symbol detection by correlation filters: the matched filter.** As we can see from Fig. 14.16, the received data signal passes through the loop filter in the  $I$  branch of the Costas loop. Assume for the moment that the data signal has not been lowpass filtered at the transmitter, so it originally has a square-wave shape. The transients of the data signal are now flattened by the loop filter within the receiver. To decide whether an incoming symbol is a logical 1 or 0, the receiver must sample the demodulated signal at the right time. But what is the “right time”? Taking a sample at the beginning of the symbol would certainly be wrong, since the filtered symbol must first be allowed to settle to its final amplitude (for example, +1 for logical 1, or -1 for logical 0). Hence, it would be more appropriate to sample that signal at the end of the symbol. The receiver then would decide that the symbol is 1 if the final amplitude is positive, or a 0 if it is negative. But what if there is noise on the signal? If the current incoming symbol is a 1 and a noise spike appears just at the end of the symbol causing the signal to take on a negative value, the receiver would falsely decide for a 0. Had it taken a sample just prior to that noise spike, the result would probably have been correct. This leads us to the idea that a decision for a symbol could be more meaningful if the receiver would not only check one single sample of the received symbol, but would rather form some statistic (for example, an average or a correlation) from all samples of the symbol waveform within the symbol period. It has been shown that the best estimate—in the statistical sense—is obtained from a *correlation filter*, which is also referred to as a *matched filter*.<sup>20</sup>

**What type of correlation filter do we need?** The correlation filter has many variants. First of all, the correlation can be performed with the undemodulated or with the demodulated symbol waveform. Correlating the undemodulated



**Figure 14.17** A block diagram of a correlation receiver. The incoming symbol is correlated with the template functions stored in the receiver.

waveform is simpler to explain, hence we start with this topic. [Figure 14.17](#) demonstrates the operating principle. It is assumed that the data signal has not been filtered at the transmitter, and that a logical 1 is represented by one full cycle of a sine wave (solid curve at top left) having initial phase 0. A logical 0 is also represented by one cycle of the sine wave, but has initial phase  $\pi$  (dashed curve). Of course, the symbols could have longer duration, so one symbol could extend over a number of sine wave cycles. The symbol waveform is sampled at the input of the correlation receiver; here it was assumed the sampling rate is four times the symbol rate. We first consider the case where the incoming symbol is a 1 (solid curve). The samples are digitized by an ADC which is not shown in the diagram. The digitized samples are shifted into a FIR filter having length 4. To be more general, assume that the four samples of the symbol have the values  $[a \ b \ c \ d]$ ; these values form a vector. The coefficients of the upper FIR filter have the same values by definition, but in reverse order, so they take on the values  $[d \ c \ b \ a]$ . When the samples are shifted into the FIR filter, after four shifts the first sample (a) will be at the last tap of the filter (coefficient a). The second sample (b) will be at the second last tap (coefficient b), and so on. At this instant, the upper filter outputs an output sample having the value  $a^2 + b^2 + c^2 + d^2$ . If there is no noise on the received symbol, this equals the autocorrelation of the

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

symbol sequence (with time delay  $\tau = 0$ ). We now consider what happens with the lower FIR filter. Its coefficients defined by  $[-d \ -c \ -b \ -a]$  hence correspond to the undistorted values of the symbol waveform for a logical 0. In our example, the lower correlator outputs the value  $-a^2 - b^2 - c^2 - d^2$  at the end of the symbol period, which is the cross correlation of the logical 1 waveform with the logical 0 waveform (with time delay  $\tau = 0$ ). Numerically, the upper correlator delivers an output of +2 when a 1 is received, and the lower correlator delivers -2. If a logical 0 is received, the reverse happens: the upper correlator outputs -2, and the lower outputs +2. To decide whether a symbol is 0 or 1, the receiver compares the two correlator outputs at the end of each symbol. Whenever the upper correlator output is more positive than the lower, the receiver decides that the symbol is a 1 and vice versa. The benefit of the correlation filter becomes obvious when we assume that a logical 1 was received, but the second sample (nominal value 1) was corrupted by noise and has a value of 0, for example. The upper correlator then would output +1, and the lower -1, which means this error does not lead to a wrong decision. An error occurs only if the noise level is so high it reverses the polarities of the correlator output signals.

In this example, the waveforms for logical 0 were chosen to be identical with the waveform for logical 1, but with inverted polarity. In such simple cases, the receiver does not need two correlators. The upper correlator in Fig. 14.17 would be sufficient: decisions would be made by looking only at the sign of the correlator output. The filters shown in the diagram are called *matched filters* because their coefficients exactly *match* the samples of the undistorted symbol waveform(s), but are arranged in reverse order. It can be shown that the decision made by a correlation receiver is a *maximum likelihood* decision in the statistical sense,<sup>20</sup> since the receiver compares the incoming symbol waveform with all existing theoretical symbol waveforms and checks which of the stored templates has the greatest similarity (likelihood) with the received symbol.

**What about raised cosine filters?** As stated, the described correlation method shown works best with carriers modulated with a square-wave data signal. To reduce the bandwidth, modern digital communications mostly work with filtered data, as discussed in Sec. 14.7.1. When considering the working principle of the Costas loop (refer again to Fig. 14.16), we became aware that the output of the loop filter in the *I* branch provides the recovered data signal. When filtering the symbol stream in the transmitter, we must design the filter such that no ISI can occur. The raised cosine filter has been considered an optimum solution for this problem. However, now in the Costas loop the data signal passes through an additional lowpass filter. As was demonstrated by the correlation receiver in the example of Fig. 14.17, the quality of the symbol decision in the receiver would be optimum when using the same RCF in place of the *I* branch of the Costas loop. Unfortunately, this does not work in the desired way, because now the signal goes through two cascaded raised cosine filters. The frequency response of the combined filter then would no longer be symmetrical around half the symbol rate, and ISI would be generated. The frequency response of the

lowpass filter in the receiver should be chosen such that the combined filter represents an RCF in order to realize minimum ISI. Theoretically, this goal is achieved only if the loop filter is an ideal lowpass filter. In the best case, we could try to approximate a brickwall filter, but this is suboptimal and results in a FIR filter with a very great number of taps. Moreover, the brickwall filter would not be matched to the symbol waveform, hence it would have poor noise suppressing capabilities.

There is a more elegant solution: the *root raised cosine filter (RRCF)*. The frequency response of the root raised cosine filter is defined to be the square root of the frequency response of the raised cosine filter. Thus, the RRCF has a frequency response of the kind

$$H(f) = \begin{cases} 1 & \text{for } |f| < \frac{1-r}{2T} \\ \cos \frac{\pi}{4} \frac{2T|f| + r - 1}{r} & \text{for } \frac{1-r}{2T} \leq |f| \leq \frac{1+r}{2T} \\ 0 & \text{for } |f| > \frac{1+r}{2T} \end{cases} \quad (14.21)$$

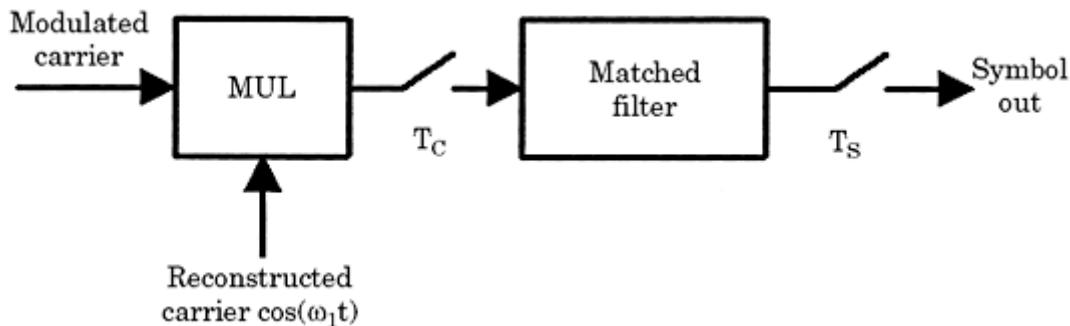
The impulse response  $h(t)$  of the RRCF is given by<sup>41</sup>

$$h(t) = \frac{\sin\left[\frac{\pi t}{T}(1-r)\right] + \frac{4rt}{T}\cos\left[\frac{\pi t}{T}(1+r)\right]}{\frac{\pi t}{T}\left[1 - \left(\frac{4rt}{T}\right)^2\right]} \quad (14.22)$$

Similar to the impulse response of the RCF, the impulse response given by Eq. (14.22) also has singularities. For some values of time  $t$ ,  $h(t)$  becomes a division 0/0. In analogy, this problem is mastered by applying L' Hôpital's rule and replacing the numerator and denominator by their derivatives.

The frequency response of the RRCF is no longer symmetrical around half the symbol rate, hence it leads to ISI, which, of course, is not desired. But if the receiver also contains an identical RRCF, the overall frequency response is identical with the RCF; thus, after the demodulator, ISI becomes zero again. The second goal—a maximum likelihood demodulator—is achieved simultaneously: the impulse response of the RRCF in the receiver closely matches the symbol waveform, which is nothing more than the impulse response of the RRCF in the transmitter. Consequently, the RRCF in the receiver is a *matched filter*. The operating principle of the matched filter is sketched by Fig. 14.18. This schematic can be considered part of the block diagram of the Costas loop in Fig. 14.16. The modulated carrier is first fed to the input of a multiplier, which serves as a demodulator. The multiplier can be an analog circuit, though in many cases a digital multiplier is used today. The matched filter is given by the RRCF. This filter is usually clocked with a frequency  $f_C$ , which is an integer multiple of the symbol rate  $f_S$ . (In Fig. 14.18,  $f_C = 1/T_C$  and  $f_S = 1/T_S$ .) The output of the matched

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

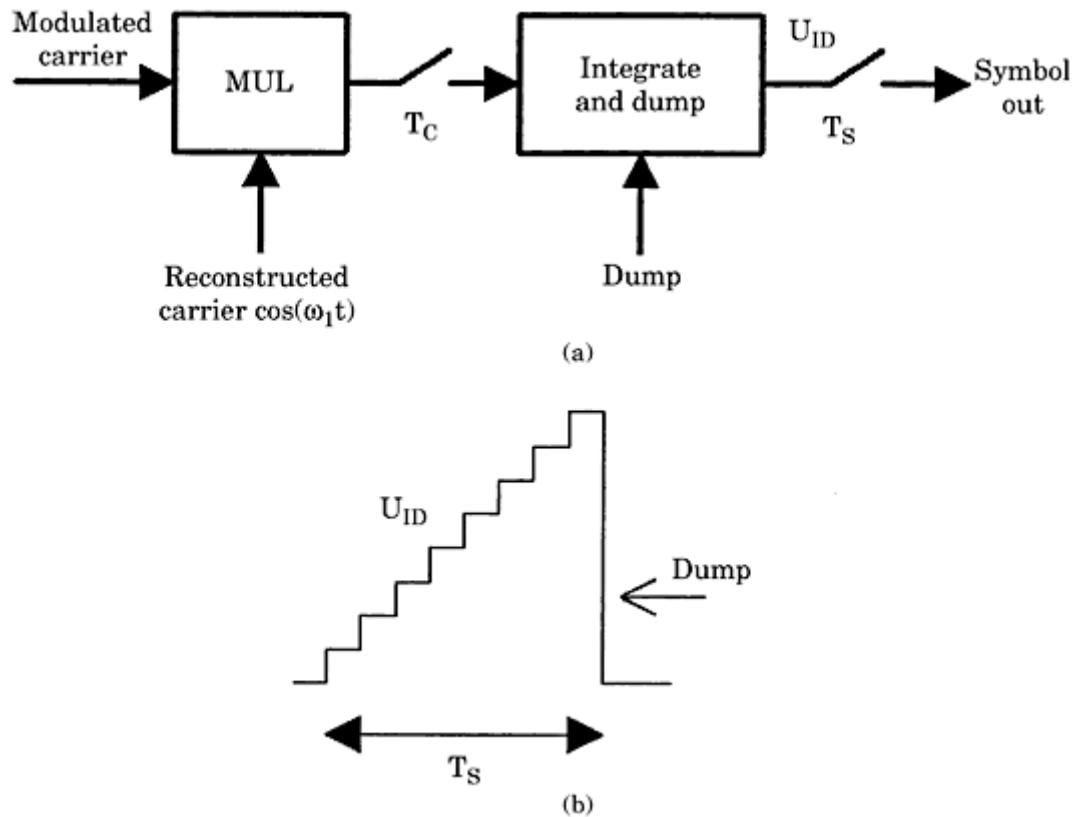


**Figure 14.18** A principle of matched filter that correlates the demodulated symbol waveform with a template stored in the receiver.

filter (correlator) is read out at the symbol rate  $f_S$ . It must be emphasized that the data signal at the output of the matched filter is no longer given by pulses or square waves, but by the superposition of oscillating waveforms, as was shown by Fig. 14.14. To reconstruct the data correctly, it is very important that the receiver samples the matched filter output at the correct times—in other words, precisely at  $t = 0, T, 2T \dots$ . This implies that the receiver must know when a symbol starts and when it is over. This must be accomplished by appropriate symbol synchronization circuits (to be discussed at the end of this section).

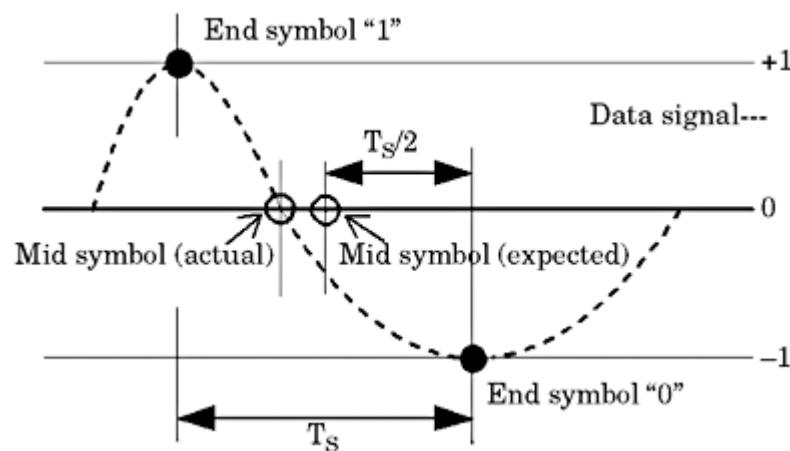
**The “integrate and dump” circuit.** When the data signal was not lowpass filtered at the transmitter, the demodulated symbol stream represents nearly a square-wave signal. The input to the matched filter then would be almost constant during the symbol period (+1 or -1). Under this condition, the correlator can be built from an integrator, as shown in Fig. 14.19a. At the end of the preceding symbol, the integrator is reset (dumped) to zero. When the next symbol is a logical 1 and is shifted into the integrator, its output builds up a staircase [cf. Fig. 14.19b]. The staircase function reaches its maximum at the end of the symbol. If the incoming symbol was a logical 0, the staircase would run negative. The symbol is determined at the end of the symbol period. Because the integrator is dumped after every symbol period, this configuration is also called the *integrate and dump circuit*. Integrated digital Costas loops such as the Harris HAS50210<sup>42</sup> provide both RRCF and integrate and dump circuits. They can be configured individually according to a given application and symbol format.

**Symbol synchronization.** To conclude, we will deal with the problems related to symbol synchronization. When performing phase synchronization, the receiver reconstructs a replica of the carrier which is locked in frequency and phase to the incoming carrier. When synchronizing with symbol timing, the receiver must re-create the symbol clock in order to know exactly when the end of a symbol period has been reached. We remember that the receiver must sample the output of the correlation filter at the instant where the incoming symbol is terminated and the next symbol period is starting. Many methods can achieve symbol synchronization; we discuss only the most often used.



**Figure 14.19** An integrate and dump circuit: (a) a block diagram; (b) output of the integrator for a square-wave input.

**“Mid-symbol” sampling.** We first discuss a method called mid-symbol synchronization. It is mainly used when the data are lowpass filtered—for example, by the root raised cosine filter. The correlator output is then sampled at the end of each symbol period (which corresponds to the instants  $t = -2T, -T, 0, T, 2T$ , and so on, in Fig. 14.20). If a sequence of 1s are received, the end-symbol



**Figure 14.20** A symbol synchronization using mid-symbol values (cf. text).

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

amplitudes will all be around +1. If a sequence of 0s is transmitted, all succeeding end-symbol amplitudes will be around -1. Whenever a 1 is following a 0 or vice versa, however, the data signal crosses zero; this is shown in Fig. 14.20. When the symbol clock is properly adjusted, the zero crossing will be in the center between two succeeding decision points (labeled "end-symbol 1" and "end-symbol 0" in the figure). The expected mid-symbol time is now calculated by taking the average of the two end-symbol times. The theoretical location of the zero-crossing is labeled "mid-symbol (expected)." If the actual zero crossing of the symbol signal deviates from this precalculated value (cf. the point labeled "mid-symbol [actual]" ), the symbol timing is erroneous and must be corrected for. The timing error (the difference between the actual and expected location of the mid-symbol) is used to adjust the frequency of a local symbol clock generator. The mid-symbol synchronization makes use of zero crossings. No timing information is available when a logical 1 is following by another logical 1. The same holds true for a sequence of 0s.

**The early-late gate.** Another symbol synchronization technique that is widely used is the *Early-Late Gate* (see Fig. 14.21). This method works best when the data are square-wave signals. In this case, the symbol signal  $s(t)$  is constant and positive during a symbol period if the incoming symbol is a 1, or constant and negative if the symbol is a 0. The "early gate" is a gated integrator summing up the symbol during the first half of the symbol period. The "late gate" is another integrator that is gated on during the second half of the symbol period. When symbol timing is correct, the outputs of both integrators are identical, as shown in Fig. 14.22a (cf. the shaded areas). The output signals  $y_1$  and  $y_2$  are equal then, and the error signal  $e$  in Fig. 14.21 is zero. When there is an offset in symbol timing, the outputs of the integrators become unequal (Fig. 14.22b), and the

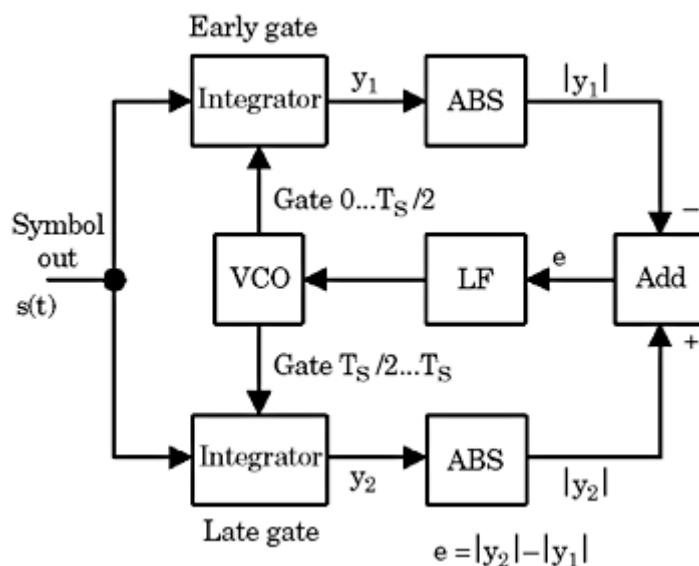
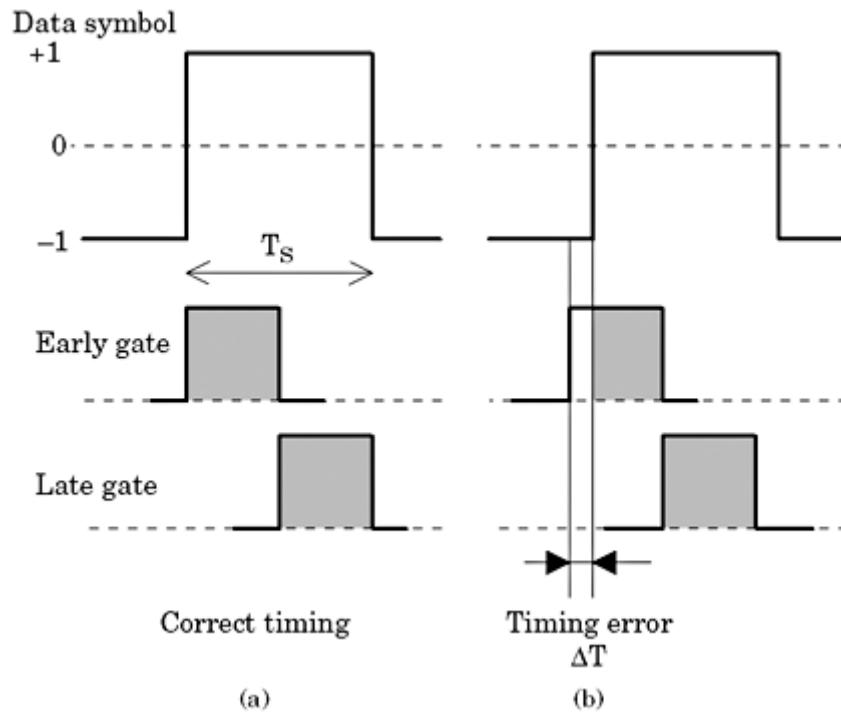


Figure 14.21 Symbol synchronization using the Early-Late Gate technique.



**Figure 14.22** Gate timing of the Early-Late Gate circuit. (a) Waveforms for correct timing.  
(b) Waveforms for timing error  $\Delta T$ .

error signal becomes positive or negative. Its polarity depends on the sign of the timing error. The waveforms in Fig. 14.22 are shown for an incoming symbol representing logical 1. If the symbol is a 0, the polarities of the integrator outputs get inverted. Because the sign of the error signal must not change when the symbol switches from 1 to 0, absolute value circuits follow the integrators (Fig. 14.21).

Additional symbol synchronization circuits have been discussed in references.<sup>1,20</sup>

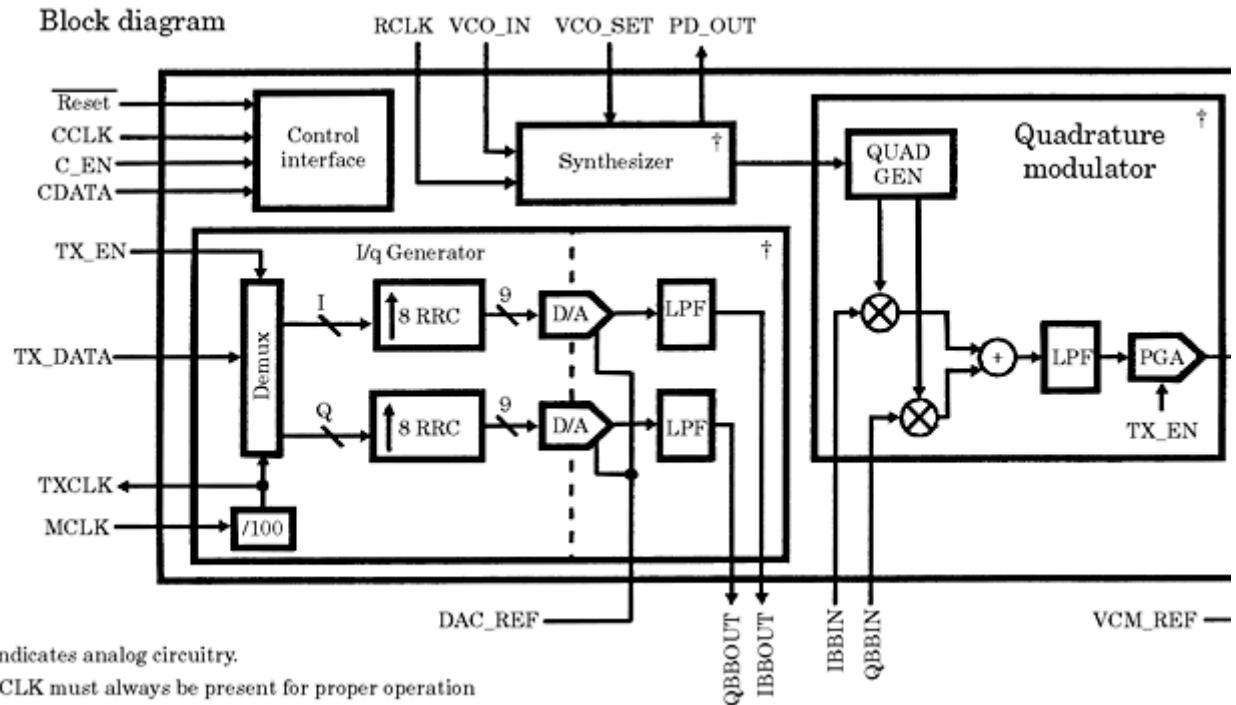
## Digital Communications Using QPSK

### Transmitter considerations

QPSK has been defined in Sec. 14.3.2, refer also to Fig. 14.3. To implement QPSK, two carriers must be generated in the transmitter, which are offset in phase by  $90^\circ$ . As explained by Fig. 14.2, the symbol stream is partitioned into two data streams; one of these is used to modulate the in-phase carrier, while the other modulates the quadrature carrier. An example of a QPSK modulator is shown in Fig. 14.23. This is a block diagram of the Harris HSP50307 Burst QPSK Modulator.<sup>43</sup> Data are fed at a rate of 256 kbit/s to the TX\_DATA input. The partitioning into  $I$  and  $Q$  streams is made by the demultiplexer. The bit rate after partitioning is reduced to 128 kbit/s.  $I$  and  $Q$  data are filtered by an RRCF. This filter has order 64 (length 65) and is clocked at a frequency of 1.024 MHz—in other words, the filter input signal is oversampled by a factor of 8. Oversampling is accomplished by zero padding: seven zeroes are inserted

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.



**Figure 14.23** A block diagram of burst QPSK modulator Harris HSP50307. (Used with the permission of Harris Corporation.)

between any two succeeding samples of the  $I$  and  $Q$  data stream. The excess bandwidth of the RRRC is chosen as  $r = 0.5$ . The filtered data are applied to the inputs of two DACs. The unwanted high-frequency signals caused by D-to-A conversion are removed by two analog lowpass filters. These filter outputs (IBBOUT and QBBOUT) modulate the two carriers, as shown in the right half of the block diagram. The carrier frequency can be programmed within a range of 8 to 15 MHz in steps of 32 kHz.

This integrated circuit is a mix of analog and digital circuitry. All operations that can be made digitally are performed by digital circuits.

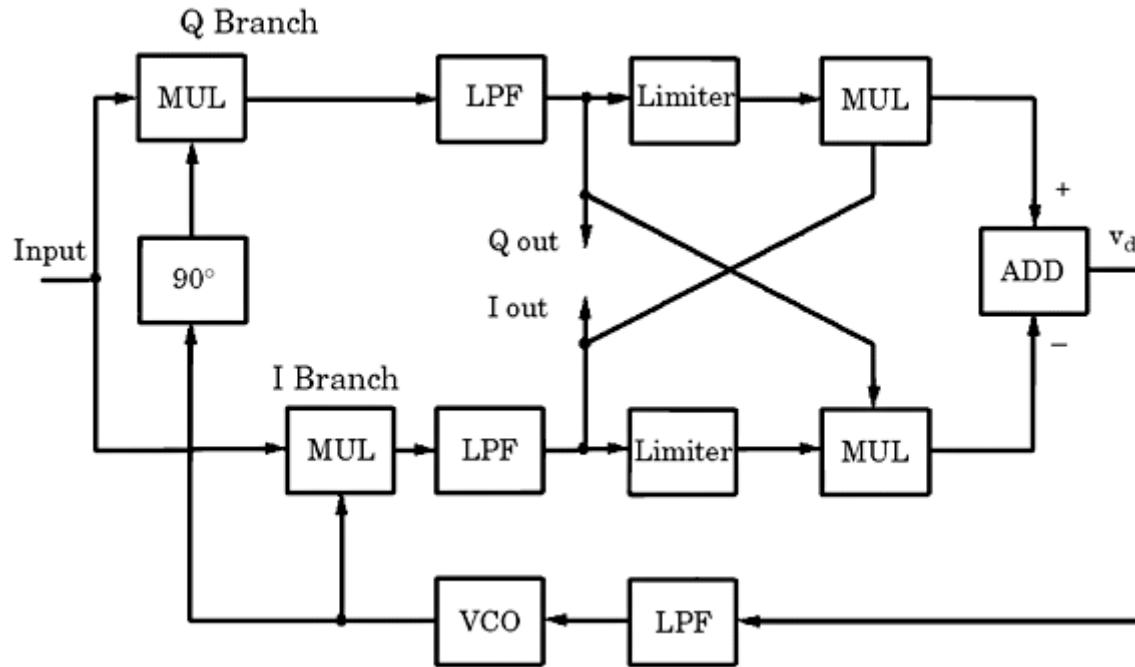
## Receiver considerations

The receiver for BPSK had to regenerate a carrier that was in phase with the carrier; this was done by the Costas loop in Fig. 14.16. With QPSK, we no longer must deal with one single carrier, but instead receive a signal that is the addition of two carriers modulated by two different data signals  $m_1(t)$  and  $m_2(t)$ , respectively. If we assume that  $m_1(t)$  modulates the in-phase carrier and  $m_2(t)$  the quadrature carrier, the combined output signal  $u_{\text{QPSK}}$  of the transmitter can be written as

$$u_{\text{QPSK}}(t) = m_1(t)\cos(\omega_1 t + \theta_1) - m_2(t)\sin(\omega_1 t + \theta_1)$$

where  $\omega_1$  is the radian frequency of the carrier and  $\theta_1$  is the phase. The Costas loop was extended to lock onto a modulated QPSK signal.<sup>1,20</sup> This circuit is shown in Fig. 14.24. It is quite similar to the original arrangement (see Fig. 14.16). The two lowpass filters (loop filters)

in the  $I$  and  $Q$  branch are now followed by limiters. By virtue of the limiters, the output signals become independent of the



**Figure 14.24** A Costas loop extended for the demodulation of QPSK.

levels of the modulating signals, so the circuit can also be used to demodulate QAM signals (cf. Sec. 14.9). When the loop has locked, the VCO will generate a signal of the form

$$u_{\text{VCO}}(t) = \sin(\omega_1 + \theta_2)$$

where  $\theta_2$  is the phase of the output signal. Following the analysis of Gardner,<sup>1</sup> the output  $v_d$  of the adder (at center right in Fig. 14.24) is given by

$$\begin{aligned} v_d(t) &= [m_1(t) \sin \theta_e + m_2(t) \cos \theta_e] \operatorname{sgn}[m_1(t) \cos \theta_e - m_2(t) \sin \theta_e] \\ &\quad - [m_1(t) \cos \theta_e - m_2(t) \sin \theta_e] \operatorname{sgn}[m_1(t) \sin \theta_e + m_2(t) \cos \theta_e] \end{aligned}$$

where  $\operatorname{sgn}$  (signum) specifies the action of the limiter and  $\theta_e = \theta_1 - \theta_2$  is the phase error. For rectangular data signals, the average DC output  $v_d$  of the adder becomes

$$v_d(t) = \begin{cases} \sin \theta_e & \text{if } -45^\circ < \theta_e < 45^\circ \\ -\cos \theta_e & \text{if } 45^\circ < \theta_e < 135^\circ \\ -\sin \theta_e & \text{if } 135^\circ < \theta_e < 225^\circ \\ \cos \theta_e & \text{if } 225^\circ < \theta_e < 315^\circ \end{cases}$$

There are four values for  $\theta_e$  where the loop can lock:  $\theta_e = 0^\circ, 90^\circ, 180^\circ$ , or  $270^\circ$ , respectively. If it locks at  $\theta_e = 0^\circ$ ,  $v_d(t) = \sin \theta_e$ , which can be approached by  $\theta_e$  for a small phase error. The Costas loop then operates like a conventional PLL.

---

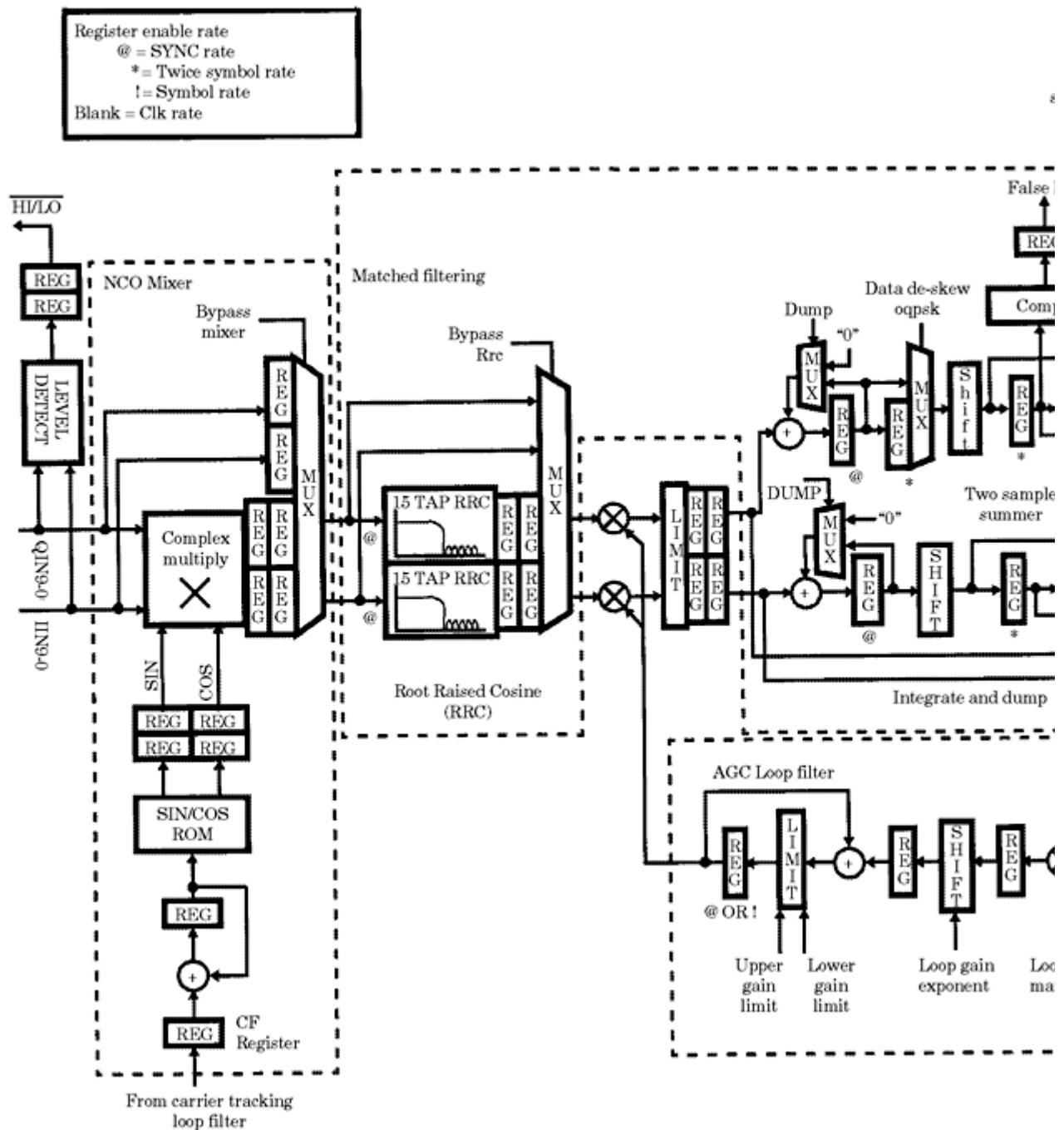
Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

If the loop locks at  $\theta_e - 90^\circ$ ,  $v_d(t)$  becomes  $-\cos \theta_e$ , which is near zero. When the phase error deviates only slightly from  $90^\circ$ ,  $v_d(t)$  is proportional to that deviation, and the circuit again operates like a simple PLL. Analog considerations can be performed with the remaining values of phase error where the Costas loop can lock.

A realization of the Costas loop for QPSK reception is found in the circuit HSP50210 manufactured by Harris.<sup>42</sup> Figure 14.25 shows a block diagram of this device. Actually, the full receiver is made up from cascading two integrated circuits—the HSP50110 (Digital Quadrature Tuner) and the HSP50210—to be described here. Demodulation of the  $I$  and  $Q$  signals take place at the complex multiply circuit at the left. “Complex multiplication” simply means that the incoming signal is multiplied with a cosine wave in one arm and with a sine wave in the other arm of the demodulator. The complex multiplier output delivers the unfiltered baseband data. These are fed to two root raised cosine filters. The sine and cosine waveforms required for demodulation are taken from a table look-up ROM, shown at the bottom left. These signals are generated digitally by interpolating values of sine and cosine functions stored in a ROM. Also shown are two integrate and dump circuits that can be used when the data signals are rectangular. More detailed information is available from application notes.<sup>45,46</sup>

## Digital Communications Using QAM

As was demonstrated in Sec. 14.5, QAM can be considered an extension to QPSK. While all symbols had the same amplitude with QPSK, the amplitude of both in-phase and quadrature components now can take on a number of different values. The circuits for transmission and reception of QAM signals are similar to those used with QPSK, but the receiver must be equipped with additional level discriminators. The major benefit of QAM is increased symbol throughput without increasing the bandwidth. A nice application of QAM is found in DVB (Digital Video Broadcasting). In Europe, a proposal for digital video broadcasting was presented under the name DVB-C.<sup>47</sup> In analog video broadcasting, channel spacing is standardized to 8 MHz. It was one of the goals of this standard to use the same bandwidth with digital TV broadcasting. This can only be realized by extensive use of source and channel coding.<sup>20</sup> Source coding is a means to reduce the symbol rate at the data source. An analog TV signal has a bandwidth of 5 MHz approximately. When transmitting the signal digitally, we therefore would have to sample it at 10 MHz. Using eight bits per pixel, the luminance signal alone needs a channel capacity of around 80 Mbit/s. When the color signal is added, we easily end up with 100 Mbit/s. To transmit the signal using QPSK, the bandwidth would become around 50 MHz which is far from realistic. By source coding, some redundancy is removed from the signal. Application of the MPEG-2 standards reduces the bit rate to about 38 Mbit/s. Because MPEG-2 adds little error protection, the redundancy of the compressed signal is slightly increased by making use of



**Figure 14.25** Ablock diagram of the Digital Costas Loop Harris HSP50210. (Used with permission

Reed-Solomon codes.<sup>20</sup> RS(188,204) is recommended in the proposal. This represents a Reed-Solomon code that adds 16 check bytes for each block of 188 bytes: instead of transmitting 188 bytes, we transmit  $188 + 16 = 204$ , hence the code RS(188,204). With this amount of added redundancy, the bit rate goes up to 41.4 Mbit/s. Using QAM<sub>64</sub>, six bits per symbol can be transmitted (refer to Fig. 14.23). Consequently, the symbol rate becomes  $41.4 \text{ Mbit/s} / 6 = 6.9 \text{ Mbaud}$ . When transmitting the signal in the baseband, a one-sided bandwidth of  $6.9/2 = 3.45 \text{ MHz}$  would be required. Because the proposal recommends the RRCF (with  $r = 0.15$ ) for filtering the data, the bandwidth is slightly increased to  $(1 + r) 3.45 = 3.97 \text{ MHz}$ . When modulating a high-frequency carrier with that signal, its two-sided bandwidth becomes  $2 \cdot 3.97 = 7.94 \text{ MHz}$ , which is below the required value of 8 MHz.

## Digital Communications Using FSK

### Simple FSK decoders: easy to implement, but not effective

FSK is still one of the most widely used techniques in the communication of digital signals. Binary FSK has been extensively used in modems for moderate speeds. In such applications, a linear, digital, or all-digital PLL can immediately serve as an FSK demodulator. Such circuits are very easily implemented, but have the disadvantage that the maximum symbol rate stays far below the limits predicted by theory. Before discussing alternatives, let's recall what should be achieved theoretically, and why the simpler circuits cannot reach these goals.

As stated in Sec. 14.4.1, there are two basic types of FSK decoding: coherent and noncoherent. Equation (14.1) shows that for coherent FSK each frequency representing a symbol value must be an integer multiple of half the symbol rate, where the symbol rate is given by  $1/T_S$ . The difference between any two frequencies used in FSK must therefore be at least half the symbol rate. If a binary coherent FSK system with a symbol rate of 2400 bit/s is envisaged, we could use the frequencies  $f_1 = 1200 \text{ Hz}$  and  $f_2 = 2400 \text{ Hz}$  to represent the binary values 0 and 1, respectively. A binary 0 would then be given by just one half cycle of a 1200 Hz oscillation, and a binary 1 would be given by one full cycle at 2400 Hz. If an LPLL, DPLL, or ADPLL were used as an FSK decoder (cf. Fig. 11.21, for example), it must lock onto a new frequency whenever an incoming symbol differs from the preceding one. Although the ADPLL is the fastest circuit of those mentioned, it would certainly not be able to settle to another input frequency within one half cycle only. The decoder circuit shown in Fig. 11.21 sets the D flipflop on every positive edge of the input signal  $u_1$ , hence it needs at least one cycle to react to a frequency step. To accomplish the required symbol rate, the frequencies  $f_1$  and  $f_2$  must be chosen much higher. This would result in a much larger bandwidth, of course.

As will be demonstrated later, a coherent detector for FSK must know the phase of the FSK signal. This requirement is dropped if the receiver uses noncoherent

detection. For noncoherent detection the minimum frequency spacing is larger by a factor of 2—in other words, the minimum spacing becomes equal to the symbol rate  $1/T_S$ . This can be explained by looking at the spectrum of the tones. We assume that FSK<sub>2</sub> is applied and two tones having frequencies  $f_1$  and  $f_2$  are used to represent binary 0 and 1, respectively. Tone 1 is called  $s_1(t)$  and is represented by a burst of a cosine wave of frequency  $f_1$  and duration  $T_S$ , hence it can be written as

$$s_1(t) = (\cos 2\pi f_1 t) \operatorname{rect}(t/T_S) \quad (14.23)$$

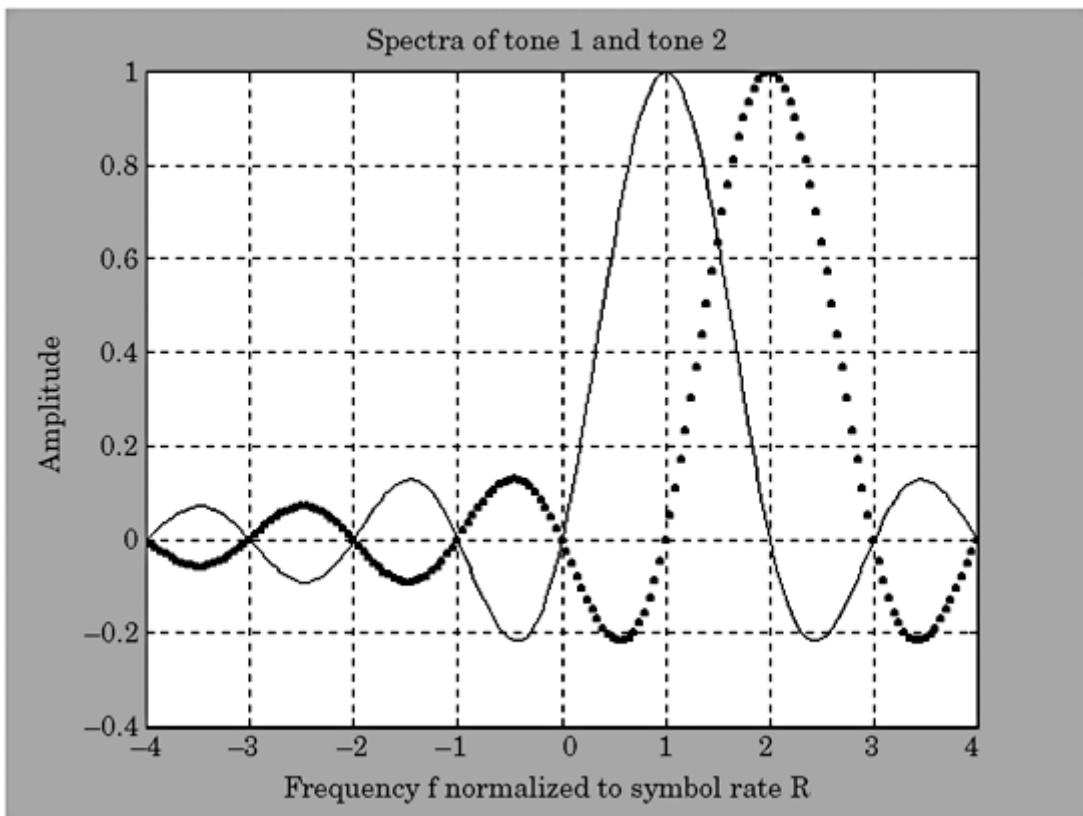
where  $\operatorname{rect}(t/T_S)$  denotes a rectangular pulse of duration  $T_S$  and is defined by

$$\begin{aligned} \operatorname{rect}(t/T_s) &= 1 && \text{for } -T_s/2 \leq t \leq T_s/2 \\ &= 0 && \text{for } |t| > T_s/2 \end{aligned}$$

The spectrum of tone 1 is a sinc function

$$S_1(f) = T_s \frac{\sin(\pi T_s(f - f_1))}{\pi T_s(f - f_1)} \quad (14.24)$$

The spectrum is symmetrical about the frequency  $f = f_1$ , as shown in Fig. 14.26 (solid curve). It has zeroes at frequencies  $f = f_1 \pm k/T_s$ , where  $k$  is an integer  $> 0$ .



**Figure 14.26** Spectra of tone 1 and tone 2 in noncoherent FSK detection.

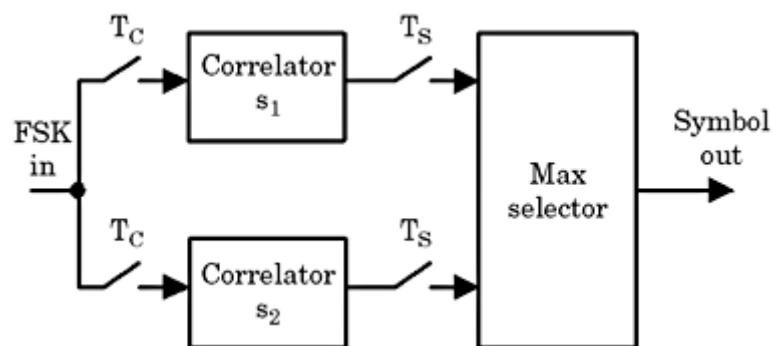
---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

In order that two tones in binary FSK not interfere with each other during detection, the peak of the spectrum of tone 1 must coincide with one of the zero crossings of the spectrum of tone 2. This requirement is fulfilled if the frequency of tone 2 is displaced by exactly  $1/T_S$ . The spectrum of tone 2 is shown by the dotted curve in Fig. 14.26. Its peak is at a location where the spectrum of tone 1 is exactly 0. Hence, the two tones can be detected without interference. This proves the frequency spacing must be equal to the symbol rate  $1/T_S$  and is therefore larger by a factor 2 than the spacing required for coherent detection. We will now discuss some of the most familiar coherent and noncoherent FSK decoder schemes.

### Coherent FSK detection

A coherent binary FSK decoder is shown in Fig. 14.27. The input is given by the demodulated FSK signal. This demodulator is not shown in the block diagram. The incoming symbol is correlated by two correlators. These are usually implemented by FIR filters; their coefficients are identical with the samples of the corresponding symbol waveforms [cf. Fig. 14.7b]. The FIR filters operate at the clock frequency  $f_C = 1/T_C$ , which is mostly an integer multiple of the symbol rate. The correlator outputs are sampled at the symbol rate  $f_S = 1/T_S$  at the end of each symbol. If  $s_1(t)$  is the waveform for a logical 0 and the incoming symbol is a logical 0, the upper correlator delivers a large positive output at the end of the symbol; the result is actually the autocorrelation of the signal  $s_1(t)$  at a delay of 0. The lower correlator then outputs a signal that is close to 0 (actually, it is the cross correlation between  $s_1(t)$  and  $s_2(t)$ , which is 0 by definition). To decide whether an incoming symbol is a 0 or a 1, the detector compares the outputs of both correlators. If the output of the upper correlator is larger than the output of the lower, the decoder decides that the symbol was a 0, and vice versa. In mary FSK, more than two frequencies are used to represent m-ary symbols—for example, four. For arbitrary m, m correlation filters must be provided in the FSK decoder.



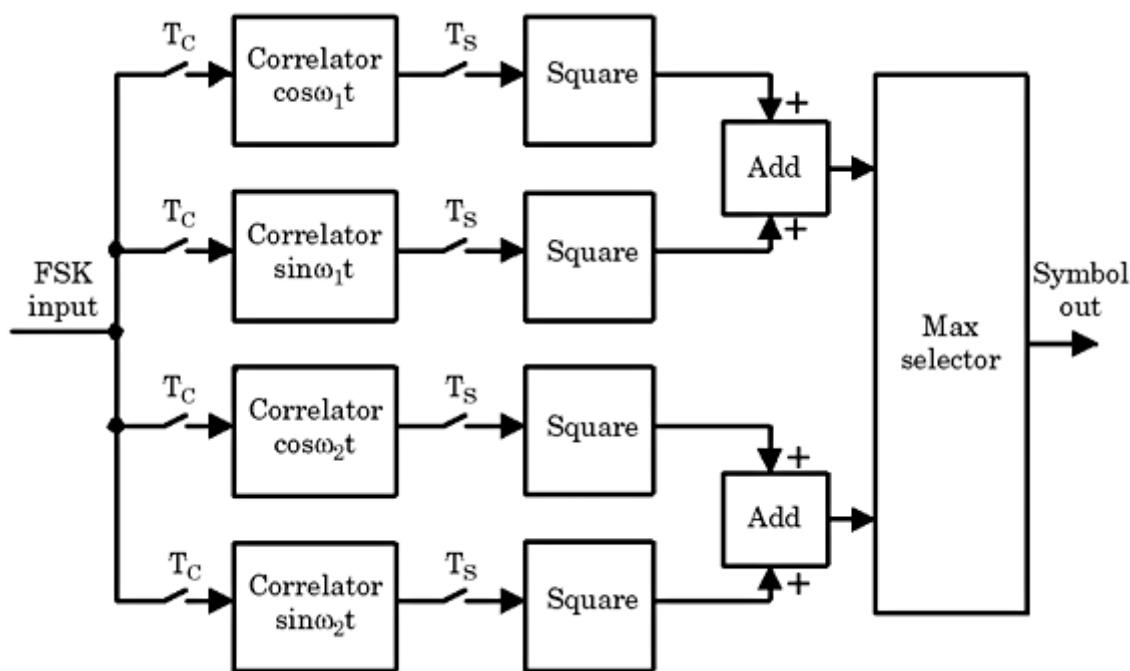
**Figure 14.27** A coherent FSK decoder (cf. text).

## Noncoherent FSK detection and quadrature FSK decoders

A noncoherent binary FSK decoder is shown in Fig. 14.28. With noncoherent detection, the receiver does not know the phase of the symbol waveforms. It therefore generates *quadrature signals* for every existing symbol waveform. This decoder is therefore referred to as a *quadrature FSK* decoder. Assume that the FSK signal can have the two radian frequencies  $\omega_1$  and  $\omega_2$ . Consequently, the decoder provides four waveforms:  $\cos \omega_1 t$ ,  $\sin \omega_1 t$ ,  $\cos \omega_2 t$ , and  $\sin \omega_2 t$ . It is still assumed that radian frequency  $\omega_1$  represents a logical 0, and radian frequency  $\omega_2$  represents a -logical 1. If the incoming symbol is a 0, its waveform can be written as

$$s_1(t) = \cos(\omega_1 t + \varphi)$$

where  $\varphi$  is an arbitrary phase. If  $\varphi$  were close to 0, there would be a strong correlation of the FSK signal with  $\cos \omega_1 t$ , so the uppermost correlator would deliver a large positive output at the symbol end, but the second correlator output would be near 0 because there is almost no correlation with the sine wave  $\sin \omega_1 t$ . If  $\varphi$  were close to  $\pi/2$ , the reverse would happen: the uppermost correlator output would be near 0, but the second correlator would deliver a large correlation. For arbitrary phase  $\varphi$ , there would be a correlation with both cosine and sine waves. By squaring and adding the outputs of the upper two correlators, the combined correlator output becomes independent of phase  $\varphi$ . In fact, the output signal of the adder is proportional to the energy of the symbol waveform. The lower two correlators provide correlation of the incoming symbol with the stored replicas for a logical 1. Again, a maximum selector checks which of the summed correlator outputs is greater at the symbol end. If it happens that the upper correlator had a larger output, the decoder decides that the received



**Figure 14.28** A noncoherent (quadrature) FSK decoder (cf. text).

---

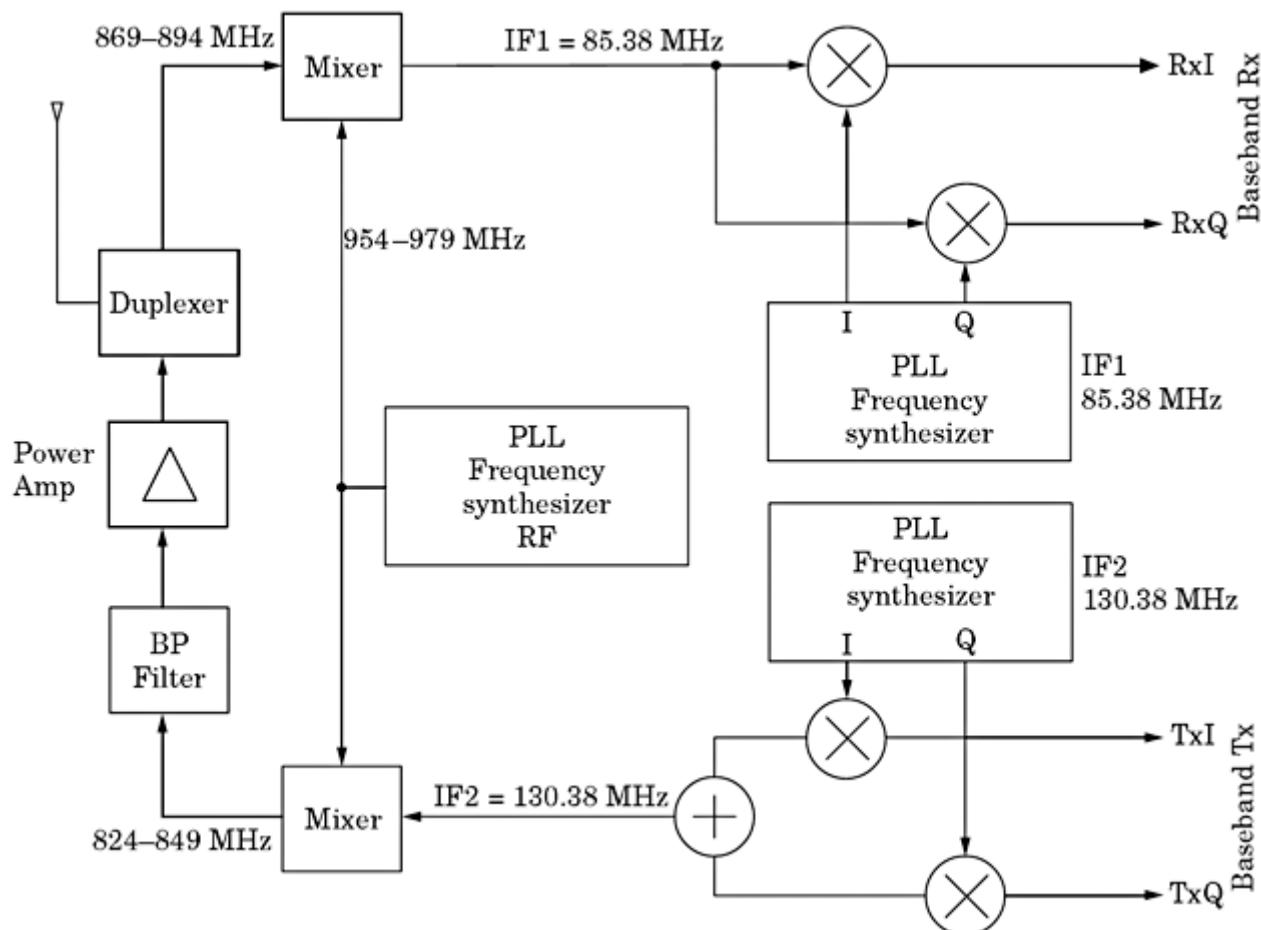
Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

symbol was a logical 0. This circuit can also be extended for m-ary FSK. It then must provide a pair of correlators (cosine and sine waves) for each state the symbol can take on.

## Digital Communications in Mobile Phones

Mobile communications make extensive use of PLL frequency synthesizers. In this section, an example of a cellular chip set is presented (see Fig. 14.29).

The upper half of the diagram is the receiver section, while the lower half is the transmitter section. The received signal covers the frequency range from 869 to 894 MHz, while the transmitted signal is in the range of 824 to 849 MHz. A PLL frequency synthesizer generates frequencies in the range of 954 to 979 MHz. This frequency band is offset from the receiving channels by 85.38 MHz, which is the intermediate frequency of the receiver (IF1). Moreover, it is offset from the transmitting channels by 130.38 MHz, which is the intermediate frequency of the transmitter (IF2). Hence, the incoming signal is mixed down to the IF1. Another PLL frequency synthesizer creates a quadrature carrier at the IF1 frequency (cf. the I and Q outputs right on top). These two carriers are used to mix down the IF1 signal to the baseband (RxI and RxQ), where it is further processed by a digital signal processor (not shown in the diagram).



**Figure 14.29** A block diagram of a mobile communication system.

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

The transmitter signal (TxI and TxQ) is mixed up to the IF2 frequency by a frequency synthesizer that creates a quadrature carrier at this intermediate frequency. The resulting IF is then mixed up to the transmitter frequency band 824 to 849 MHz.

## Bandwidth Efficiency, Bit Error Rate (BER), and the Signal Power Efficiency of Digital Modulation Schemes

It is one of the primary goals in the design of a communication link to transmit the signals with a bandwidth as low as possible. The various digital modulation schemes discussed in this chapter differ very much in bandwidth demand. A figure of merit is bandwidth efficiency  $R$  which is expressed in bits/s/Hz. An  $R$  value of 4, for example, tells us that we can transmit four bits for every hertz of bandwidth. [Table 14.1](#) lists the theoretical bandwidth efficiency of the modulation principles discussed earlier.

The numbers given here are based on the assumption that ideal filters are used in the demodulation circuits. In case of BPSK, for example, it is known that—without superimposed noise—the baseband signal can be detected error free when the received signal is lowpass filtered with an ideal filter having bandwidth  $f_s/2$ , or rather half the symbol rate. With bandpass modulation, two sidebands are generated—an upper and a lower one—hence the two-sided bandwidth becomes  $f_s$  and not  $f_s/2$ . With such an ideal filter, the bandwidth efficiency would be 1 bits/s/Hz. As we have seen, real lowpass filters have an excess bandwidth, which means the effective bandwidth can be about 35 percent larger. A more realistic bandwidth efficiency for BPSK would therefore be in the range of 0.7. The same consideration applies to all other types of modulation.

**TABLE 14.1 The Bandwidth Efficiency  $R$  for Various Digital Modulation Schemes**

Type of modulation	Type of detection	$m$	$R$ [bits/s/Hz]
MPSK	coherent	2 (BPSK)	1
		4 (QPSK, OQPSK)	2
		8	3
		16	4
MFSK	noncoherent	2	1/2
		4	1/2
		8	3/8
		16	1/4
	coherent	2	1
		4	1
		8	3/4
		16	1/2
MSK	coherent	2	2
QAM	—	16	4
		64	6

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

Next we will deal with FSK. When noncoherent detection is applied, the distance between the signal frequencies must be as large as the symbol rate. This implies that for m-ary FSK, the required bandwidth is  $m$  times the symbol rate. For  $m = 2$ , the bandwidth is  $W = 2f_s$ , and because only one bit is transmitted per symbol, the theoretical bandwidth efficiency becomes one half. For quaternary FSK ( $m = 4$ ), the required bandwidth becomes  $W = 4f_s$ , and because two bits per symbol are transmitted, the bandwidth efficiency is one-half again. For  $m = 8$ , however, the required bandwidth is  $W = 8f_s$ , and the efficiency drops to  $3/8$ , and so on.

The theoretical limit is improved by a factor of 2 when the FSK signal is coherently detected. This stems from the fact that the spacing of signal frequencies can now be as low as half the symbol frequency.

For MSK, the theoretical minimum bandwidth is as low as  $f_s/2$  because the carrier frequency  $f_0$  is symmetrically frequency modulated by  $\pm f_s/4$ . This yields a theoretical efficiency of 2 bits/s/Hz. Because ideal filters are not realizable, the effective numbers are lower; there are reported figures of 1.35 bits/s/Hz for GSM digital cellular.

By far the highest bandwidth efficiencies are obtained with QAM. The table does not tell, however, that such figures are only obtained at the expense of signal power. In every communication link, noise is present. An important factor to be considered is *bit error rate (BER)*. For each type of modulation, BER is a function of the ratio  $E_b/N_0$ , where  $E_b$  is the energy per bit (in Ws) and  $N_0$  is the power spectral density of thermal noise (in Ws or W/Hz).

$N_0$  is given by  $N_0 = kT$ , where  $k$  is the Boltzmann constant ( $k = 1.4 \cdot 10^{-23}$  Ws/K) and  $T$  is absolute temperature in Kelvin. For binary signals (for example, BPSK), energy per bit is defined as

$$E_b = \int_0^{T_s} s^2(t) dt \quad (14.25)$$

where  $s(t)$  is the signal representing the binary information and  $T_s$  is the duration of the bit cell.  $s(t)$  is specified in V, and if we assume that the signal is applied to a 1-ohm load resistor,  $E_b$  is the energy of this waveform in Ws. For m-ary modulation schemes, another term called *symbol energy*  $E_s$  has been defined. For arbitrary  $m$ , the symbol energy becomes

$$E_s = \text{ld } m E_b = k E_b \quad (14.26)$$

with  $k = \text{number of bits transmitted per symbol}$ ,  $k = \text{ld } m$ . The ratio  $E_b/N_0$  is usually specified in dB—in other words, we have

$$E_b/N_{0(\text{dB})} = 10 \log_{10} E_b/N_0 \quad (14.27)$$

To achieve a predetermined BER—for example,  $\text{BER} = 10^{-5}$  (which means that one in  $10^5$  bits is erroneous)—for each modulation scheme the required energy per bit (or energy per symbol) has been computed; very detailed information is

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

found, for example, in Sklar.<sup>20</sup> It turns out that 16-ary FSK  $E_b/N_{0(\text{dB})}$  is in the range of 8 dB when BER =  $10^{-5}$  is required. For 8-ary MPSK, for example,  $E_b/N_{0(\text{dB})}$  must be around 14 dB for the same BER. For 256-ary QAM, however,  $E_b/N_{0(\text{dB})}$  is required to be about 23 dB.

This clearly demonstrates that there exist modulation schemes that are highly bandwidth efficient but require large signal energy, and that other schemes exist that are much less bandwidth efficient but are able to work with much lower signal energy.

## Searching PLL Integrated Circuits

The designer of a PLL system has a broad choice of ICs built from different semiconductor technologies. A great number of ICs contain a complete PLL system. In addition, some ICs include only one or more phase comparators, a single VCO, or just an analog multiplier. Furthermore, many PLL frequency synthesizer systems are available in one single package. A broad choice is available among integer- $N$  and fractional- $N$  synthesizers. Some of these ICs contain both integer- $N$  and fractional- $N$  systems on the same chip. Moreover, a great number of PLL-based semiconductor devices have been introduced that contain the full circuitry for radio and television receivers or for mobile phones. Because the frequency range of many such circuits exceeds the capabilities of standard CMOS devices, the number of high-speed prescalers is increasing quickly.

The life cycle of PLL products has become shorter than in former times, so providing a table with these products becomes obsolete after a brief time. The Web offers very effective tools for searching PLL circuits, PLL components such as VCOs, or complete systems like frequency synthesizers. In this chapter, some of these web sites are presented. A well-known data collection is provided by IC Master and can be found at [www.icmaster.com](http://www.icmaster.com), (see Fig. 15.1).

Starting a search for PLL circuits reveals 64 manufacturers. Clicking one of the companies brings up a list of their devices. Data sheets are available in PDF format for most of the circuits. Links to the homepages of all companies are included.

Another search facility is [www.datasheets.org.uk](http://www.datasheets.org.uk) (cf. Fig. 15.2).

Data sheets in PDF format can also be downloaded here. Still another facility is provided by [www.DataSheet4U.com](http://www.DataSheet4U.com), as shown in Fig. 15.3.

These are just a few examples of how to search for ICs. Besides data sheets, many search facilities also provide application notes, tutorials, or PLL simulation and design tools. Such a design tool can be downloaded, for example, from [www.nxp.com](http://www.nxp.com), a web site maintained by the Philips company (see Fig. 15.4).

The screenshot shows the IC Master search interface. At the top, there's a header bar with a user message, a welcome message, and a logout link. Below it is the IC Master logo and a search bar with the text "SEARCH OVER 100 MILLION PARTS". The search bar includes fields for "Find Product", "Starting With" (set to "PLL"), and a "Search" button. Underneath the search bar, a message says "Showing 31-40 of 64 suppliers". A table lists 10 manufacturers with their part counts and links to their websites. The table has columns for Manufacturer, Parts Found, Parametric Search, and WebSite. The manufacturers listed are Motorola Semiconductor, Murata Manufacturing, NEC Electronics, NTE Electronics, NXP Semiconductors, National Semiconductor, New Japan Radio, Nippon Precision Circuits, Oki Semiconductor, and On Semiconductor.

Manufacturer	Parts Found	Parametric Search	WebSite
Motorola Semiconductor	233 Part(s)	Parametric	<a href="http://www.motorola.com/index.htm">http://www.motorola.com/index.htm</a>
Murata Manufacturing	4 Part(s)	Parametric	<a href="http://www.murata.com">http://www.murata.com</a>
NEC Electronics	2 Part(s)	Parametric	<a href="http://www.necel.com">http://www.necel.com</a>
NTE Electronics	21 Part(s)	Parametric	<a href="http://www.nteinc.com">http://www.nteinc.com</a>
NXP Semiconductors	142 Part(s)	Parametric	<a href="http://www.nxp.com">http://www.nxp.com</a>
National Semiconductor	Parametric (> 250 Parts)	Parametric	<a href="http://www.national.com/">http://www.national.com/</a>
New Japan Radio	18 Part(s)	Parametric	<a href="http://www.njr.co.jp/index_e.htm">http://www.njr.co.jp/index_e.htm</a>
Nippon Precision Circuits	11 Part(s)	Parametric	<a href="http://www.npcamerica.com">http://www.npcamerica.com</a>
Oki Semiconductor	12 Part(s)	Parametric	<a href="http://www.okisemi.com/">http://www.okisemi.com/</a>
On Semiconductor	156 Part(s)	Parametric	<a href="http://www.onsemi.com/home/0_160">http://www.onsemi.com/home/0_160</a>

Showing 31-40 of 64 suppliers Prev | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

**Figure 15.1** Searching for PLL ICs with IC Master.

The screenshot shows the datasheets.org.uk search interface. At the top, there's a search bar with "VCO" and a "Search" button. Below it is a section for "Part Search" and "Description Search" with a dropdown menu set to "Contains (Slower)". There are also links for "Semiconductor Equipment" and "Photo Interrupter". The main content area shows a table of search results for "Datasheets 1 - 10 of about 1909 for VCO". The table has columns for Part Number, Description, Manufacturer, Information Type, Information, File Size, and Price Guide. The results list various Toshiba VCO parts like 1SV229, 1SV239, 1SV257, etc. At the bottom right, there's a "Results Pages: 1 | 2 | 3 | 4" link.

Part Number	Description	Manufacturer	Information Type	Information	File Size	Price Guide
1. <b>1SV229</b>	VCO for UHF band radio	Toshiba	Datasheet	<a href="#">Download PDF</a>	160.87 kB	None
2. <b>1SV239</b>	Variable capacitance silicon diode using as VCO for UHF radio	Toshiba	Datasheet	<a href="#">Download PDF</a>	159.82 kB	None
3. <b>1SV257</b>	Varactor Diode, VCO, Single, 15V, 1-1F1A, 2-Pin	Toshiba	Datasheet	<a href="#">Download PDF</a>	318.14 kB	None
4. <b>1SV270</b>	Variable capacitance silicon diode using as VCO for UHF band radio	Toshiba	Datasheet	<a href="#">Download PDF</a>	157.21 kB	None
5. <b>1SV273</b>	Varactor Diode, VCO, Single, 10V, 80D-323, 2-Pin	Toshiba	Datasheet	<a href="#">Download PDF</a>	212.36 kB	None
6. <b>1SV276</b>	Variable capacitance silicon diode using as VCO for UHF band radio	Toshiba	Datasheet	<a href="#">Download PDF</a>	152.77 kB	None
7. <b>1SV277</b>	Variable capacitance silicon diode using as VCO for UHF band radio	Toshiba	Datasheet	<a href="#">Download PDF</a>	154.78 kB	None
8. <b>1SV279</b>	Variable capacitance silicon diode, used as VCO for V/UHF band radio	Toshiba	Datasheet	<a href="#">Download PDF</a>	154.87 kB	None
9. <b>1SV280</b>	Variable capacitance silicon diode, used as VCO for UHF band radio	Toshiba	Datasheet	<a href="#">Download PDF</a>	152.53 kB	None
10. <b>1SV281</b>	Variable capacitance silicon diode, used as VCO for V/UHF band radio	Toshiba	Datasheet	<a href="#">Download PDF</a>	152.46 kB	None

Datasheets 1 - 10 of about 1909 for VCO Results Pages: 1 | 2 | 3 | 4

**Figure 15.2** Searching for ICs with [www.datasheets.org.uk](http://www.datasheets.org.uk).

---

**Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.**

**Datasheet4U.com**  
Mirror site : [www.DataSheet4U.net](http://www.DataSheet4U.net) **24/7 HIGH SPEED  
Fast Server**

DataSheet4U.com is a free electronic engineering tool that enables you to locate product datasheets from hundreds of electronic component manufacturers.

800,000's PDF DataSheet (ex) LM317

### Search Result Fractional N

<b>LVDS-Design-Ressourcen</b> Download Datenbl., Applikations-schriften, Design-Hilfen uvm. <a href="http://lvds.national.com">lvds.national.com</a>	<b>Semiconductors</b> Search Our Huge Selection of Quality Electronic Components Here! <a href="http://www.digikey.com/de">www.digikey.com/de</a>	<b>Relays, Sensors, Switches</b> Wide range of Reed Relays, Sensors and Switches for many applications <a href="http://www.meder.com">www.meder.com</a>
--	---	---

**LMX2350** - National Semiconductor  
PLLatinum **Fractional N** RF / Integer N IF Dual Low Power Frequency Synthesizer  
[www.datasheet4u.com/html/LMX/LMX2350\\_NationalSemiconductor.pdf.html](http://www.datasheet4u.com/html/LMX/LMX2350_NationalSemiconductor.pdf.html)

**LMX2352TM** - National Semiconductor  
PLLatinum **Fractional N** RF / Integer N IF Dual Low Power Frequency Synthesizer  
[www.datasheet4u.com/html/LMX/LMX2352TM\\_NationalSemiconductor.pdf.html](http://www.datasheet4u.com/html/LMX/LMX2352TM_NationalSemiconductor.pdf.html)

**LMX2353** - National Semiconductor  
PLLatinum **Fractional N** Single 2.5 GHz Frequency Synthesizer  
[www.datasheet4u.com/html/LMX/LMX2353\\_NationalSemiconductor.pdf.html](http://www.datasheet4u.com/html/LMX/LMX2353_NationalSemiconductor.pdf.html)

**LMX2353SLBX** - National Semiconductor  
PLLatinum **Fractional N** Single 2.5 GHz Frequency Synthesizer  
[www.datasheet4u.com/html/LMX/LMX2353SLBX\\_NationalSemiconductor.pdf.html](http://www.datasheet4u.com/html/LMX/LMX2353SLBX_NationalSemiconductor.pdf.html)

**LMX2353TM** - National Semiconductor  
PLLatinum **Fractional N** Single 2.5 GHz Frequency Synthesizer  
[www.datasheet4u.com/html/LMX/LMX2353TM\\_NationalSemiconductor.pdf.html](http://www.datasheet4u.com/html/LMX/LMX2353TM_NationalSemiconductor.pdf.html)

**Ads by Google**  
**Specs, Crosses & Source**  
Free Electronic Component Datasheets for Electronics Info  
[www.supplyframe.com](http://www.supplyframe.com)

**MOTOROLA DEVICES**  
Programming MOTOROLA DI/O programmer.  
[www.dataio.com/products](http://www.dataio.com/products)

**Zigbee / IEEE802.15.4**  
Semiconductor devices for low power sensor networks  
[www.jennic.com](http://www.jennic.com)

**Ads by Google**  
**srnd reed switch**  
coto reed switch srn reed switch  
[www.cashtec.com.tw](http://www.cashtec.com.tw)

Figure 15.3 A parts search using [www.DataSheet4U.com](http://www.DataSheet4U.com).

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

The screenshot shows the NXP Semiconductors website. At the top, there's a navigation bar with links for "SEARCHING PLL INTEGRATED CIRCUITS" and "Ronald E. Best". Below the header is the NXP logo with the tagline "founded by Philips". A sub-navigation bar includes "ABOUT NXP | PRODUCTS | SUPPLY CHAIN". The main content area has a breadcrumb trail: "You are here: NXP > Standard ICs > Products > Phase-Locked Loops". On the left, a sidebar titled "Standard ICs Quick Find" contains a search input field and a "GO" button. The sidebar menu includes sections like "I Want to...", "Product Information", "Standard ICs" (which is expanded to show "Logic", "I<sup>2</sup>C", "Interface", "Microcontrollers", "Analog", "RF", "Products", "Literature", "Packaging", "Support", "Quality", and "Contact Us"), "Applications", "Jobs", "News Center", "Company Profile", and "My Semiconductors". On the right, the main content area is titled "Phase-locked loops". It features a "PRODUCTS" section with a "All and New" heading and two links: "All Phase-Locked Loops" and "New Phase-Locked Loops". Below this are sections for "Function Numbers" (listing "4046 PLLs with VCO", "7046 PLLs with Lock Detectors", and "9046 PLLs with Bandgap Controlled VCO") and "MORE INFORMATION" (with a "PLL Design Assistance" section containing links to software, tool README files, and contact information). At the bottom of the page, there are links for "NXP | Privacy policy | Terms of use" and a copyright notice: "©2007 NXP Semiconductors. All rights reserved."

**Figure 15.4** A web site providing information about products, including a PLL design tool.

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

## The Pull-in Process

Because the pull-in process is a nonlinear phenomenon, it can be calculated to an approximation only. Different authors have derived expressions for pull-in range  $\Delta\omega_p$  and pull-in time  $T_p$  for one particular linear PLL—in other words, for the LPLL that contains a passive loop filter.<sup>1,2,4</sup> The approximation developed by the author is much more general and can be used to calculate  $\Delta\omega_p$  and  $T_p$  for any kind of LPLL and DPLL. Though the procedure is simpler than those presented by other authors, practical experiments have proved that the new approximations come closer to reality. As we will see, the simplified model derived to calculate  $\Delta\omega_p$  and  $T_p$  for the LPLL can be adapted to calculate these parameters for the DPLL as well, with the exception of the DPLL using a PFD. As pointed out in Sec. 3.9.3, the model in Fig. 3.19 was used to calculate the pull-in process of this kind of DPLL, so we can restrict ourselves to the DPLLs having an EXOR or a JK-flipflop phase detector. We start with the model for the pull-in process of the LPLL and later extend the method to the DPLL.

### The Simplified Model for the Pull-in Range $\Delta\omega_p$ of the LPLL

We assume that a linear PLL system (as shown in Fig. 2.1) is switched on at  $t = 0$ . The frequency  $\omega_1$  of the reference signal is furthermore assumed to be so much higher than the (scaled down) center frequency  $\omega_0'$  of the PLL that the initial frequency offset  $\Delta\omega_0 = \omega_1 - \omega_0'$  is greater than the lock range  $\Delta\omega_L$ . Therefore, the LPLL will not lock immediately, and the VCO will oscillate initially at the center frequency  $\omega_0$ . As we know from Sec. 3.9.3, the PLL will pull in when the initial frequency offset  $\Delta\omega_0$  is smaller than the *pull-in range*  $\Delta\omega_p$ . We are looking now for a method to compute  $\Delta\omega_p$  to an approximation.

In the following computation, the instantaneous (angular) frequency of the VCO is denoted as  $\omega_2$ , with  $\omega_2(0) = \omega_0$ . As long as the PLL stays unlocked, the output signal of the phase detector is an AC signal given by

$$u_d(t) = K_d \sin[(\omega_1 - \omega_2')t + \Phi] \quad (\text{A.1})$$

where  $\omega_2' = \omega_2/N$ ,  $N$  = scaling factor of an optional down counter, and  $\Phi$  is the zero phase, which is not of concern for the following derivation. To make the calculation simpler, we assume that the initial frequency offset  $\Delta\omega_0$  is much larger than the corner frequency  $1/\tau_2$  of the loop filter (refer to Fig. 2.17a). The gain of the loop filter for these high frequencies is therefore constant and is given by

$$F_H \approx \frac{\tau_2}{\tau_1 + \tau_2} \quad (\text{A.2a})$$

for the passive lead-lag filter (Fig. 2.17a)

$$F_H \approx K_a \frac{\tau_2}{\tau_1} \quad (\text{A.2b})$$

for the active lead-lag filter (Fig. 2.19a), or

$$F_H \approx \frac{\tau_2}{\tau_1} \quad (\text{A.2c})$$

for the active PI filter (Fig. 2.21a).

$F_H$  denotes “gain at high frequencies” in these formulas. Because  $\tau_1 \gg \tau_2$  in most cases, all three equations (A.2a) to (A.2c) can be replaced by the approximation

$$F_H \approx K_a \frac{\tau_2}{\tau_1} \quad (\text{A.3})$$

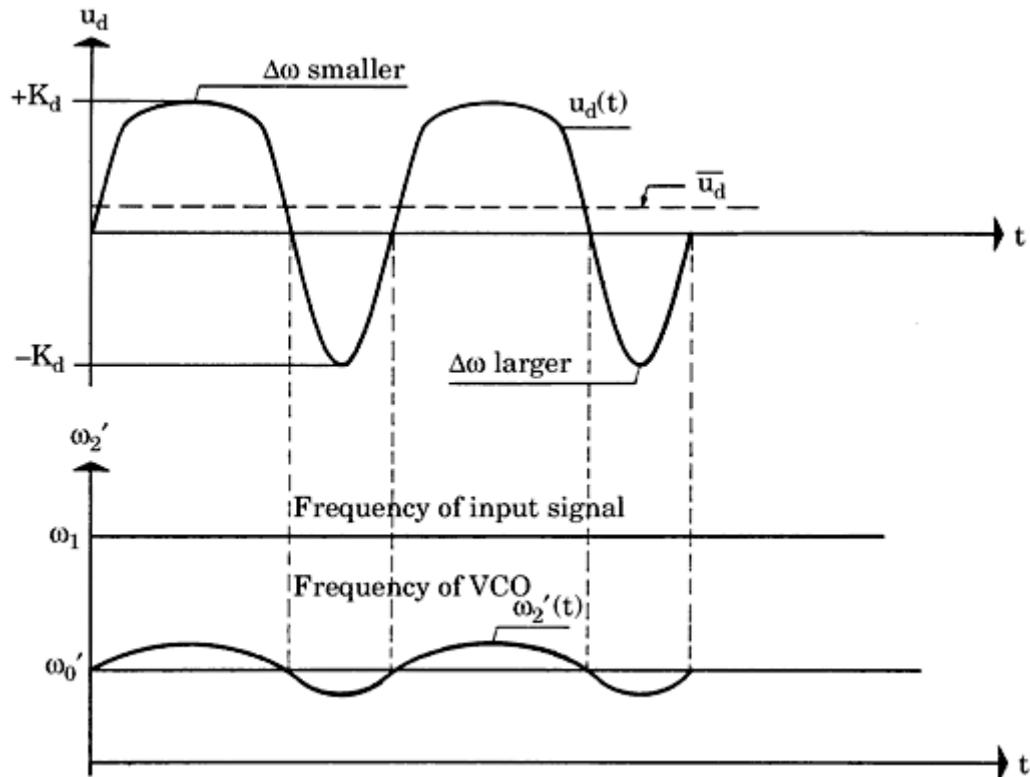
where  $K_a$  is the DC gain of the filter and must be set to 1 for the passive lead-lag and for the active PI filter. Now the output signal of the loop filter can be written as

$$u_f(t) = F_H u_d(t) = F_H K_d \sin[(\omega_1 - \omega_2')t + \Phi] \quad (\text{A.4})$$

This signal causes a frequency modulation of the VCO output signal (see Fig. A.1, bottom curve); its peak frequency deviation is given by  $F_H K_0 K_d / N$ . A closer look at this illustration reveals that the difference  $\omega_1 - \omega_2'$  between reference and output frequencies is no longer a constant but varies with the frequency modulation of the VCO signal. Figure A.1 shows that the difference  $\omega_1 - \omega_2'$  becomes smaller during the time intervals when the frequency of the VCO is increased, and vice versa. The signal  $u_d(t)$  therefore becomes inharmonic (see the top curve in Fig. A.1)—in other words, the

duration of the positive half-waves is longer than that of the negative ones. Consequently, the average value of  $\bar{u}_d$  of the phase-detector output signal is *not zero*, but slightly *positive*. This slowly pulls the frequency  $\omega_2$  of the VCO in the positive direction.

Under certain conditions to be discussed in the following, this process is *regenerative*—in other words, the down-scaled VCO output frequency  $\omega_2'$  is



**Figure A.1** A plot of  $u_d(t)$  and  $\omega_2'(t)$  against time for the unlocked PLL. It is assumed the power supply of the PLL has been turned on at  $t = 0$ .

pulled to a frequency close enough to  $\omega_1$  that the PLL finally locks. For the calculation of the pull-in process, we assume that the down-scaled frequency  $\omega_2'$  of the VCO has already been pulled somewhat in the direction of  $\omega_1$  (see Fig. A.2). The average down-scaled angular frequency of the VCO is denoted by  $\omega_{20}'$ , and the average frequency offset is  $\Delta\omega = \omega_1 - \omega_{20}'$ . Next, the average value  $\bar{u}_d$  is calculated as a function of the frequency offset  $\Delta\omega$ .

As will be demonstrated below,  $\bar{u}_d$  can be computed from the waveform of  $\omega_2'(t)$ , hence we will derive first an approximation for  $\omega_2'(t)$ . Four distinct points— $A$ ,  $B$ ,  $C$ , and  $D$ —of the signal  $\omega_2'(t)$  (cf. the lower curve in Fig. A.2) are known exactly. At points  $A$  and  $C$ , the instantaneous frequency  $\omega_2'(t)$  is exactly  $\omega_{20}'$ . At point  $B$ ,  $\omega_2'(t)$  is at its positive peak deviation, that is

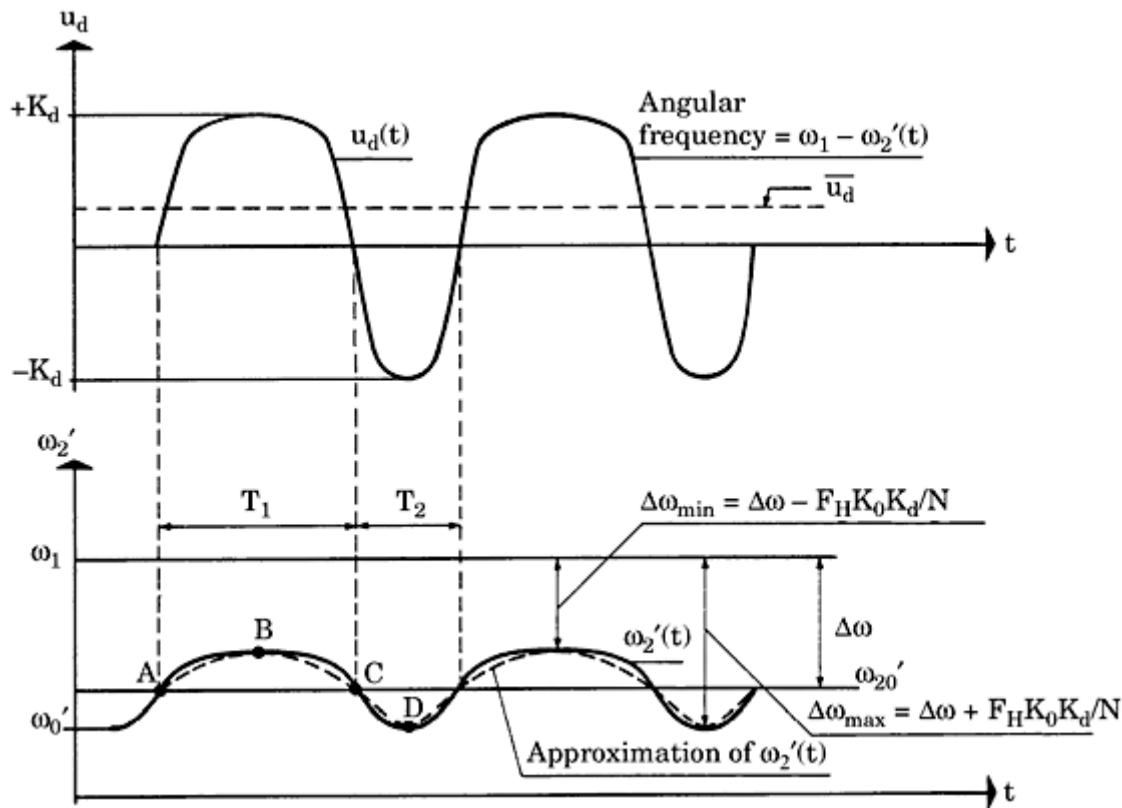
$$\omega_2'(t) = \omega_{20}' + F_H K_0 K_d / N$$

By analogy, the frequency at point  $D$  is at its minimum and is given by

$$\omega_2'(t) = \omega_{20}' - F_H K_0 K_d / N$$

The function  $\omega_2'(t)$  is now approximated by two sine half-waves (see the dashed curve in

[Fig. A.2](#)). This simplification allows us to now calculate the average frequencies  $\overline{\omega_2^+}$  and  $\overline{\omega_2^-}$  during the positive and negative half-waves, respectively. The average value of a half-wave is obtained by multiplying its peak amplitude by  $2/\pi$ .



**Figure A.2** A plot of  $u_d(t)$  and  $\omega_2'(t)$  for the unlocked PLL during the pull-in process. It is assumed that the frequency of the VCO has already been pulled somewhat toward the reference frequency  $\omega_1$ .

Thus, for  $\overline{\omega_{2+}'}$  and  $\overline{\omega_{2-}'}$  we get

$$\overline{\omega_{2+}'} = \omega_{20}' + \frac{2}{\pi} F_H K_0 K_d / N \quad (\text{A.5a})$$

$$\overline{\omega_{2-}'} = \omega_{20}' - \frac{2}{\pi} F_H K_0 K_d / N \quad (\text{A.5b})$$

The average values of the frequency offset  $\Delta\omega(t) = \omega_1 - \omega_{20}'(t)$  in the positive and negative half-waves can now be calculated as well:

$$\overline{\Delta\omega_+} = \Delta\omega - \frac{2}{\pi} F_H K_0 K_d / N \quad (\text{A.6a})$$

$$\overline{\Delta\omega_-} = \Delta\omega + \frac{2}{\pi} F_H K_0 K_d / N \quad (\text{A.6b})$$

with  $\overline{\Delta\omega_+}$  = average frequency offset in the positive half-wave and  $\overline{\Delta\omega_-}$  = average frequency offset in the negative half-wave. The duration of the half-waves  $T_1$  and  $T_2$  ([Fig.](#)

[A.2](#)) can then be approximated from  $\overline{\Delta\omega_+}$  and  $\overline{\Delta\omega_-}$ :

$$T_1 = \frac{1}{2} \frac{2\pi}{\overline{\Delta\omega_+}} = \frac{\pi}{\Delta\omega - (2/\pi)F_H K_d K_0 / N} \quad (\text{A.7a})$$

$$T_2 = \frac{1}{2} \frac{2\pi}{\overline{\Delta\omega_-}} = \frac{\pi}{\Delta\omega + (2/\pi)F_H K_d K_0 / N} \quad (\text{A.7b})$$

Knowing  $T_1$  and  $T_2$ , we can now calculate  $\overline{u_d}$ . As shown in the curve part of Fig. A.2, the amplitude of the positive half-wave of  $u_d(t)$  is  $K_d$ , and the amplitude of the negative half-wave is  $-K_d$ . Because the duration of the positive and negative half-waves is different, the average value  $\overline{u_d}$  of signal  $u_d(t)$  becomes nonzero. As we did for the computation of the  $\omega_2'$  waveform, we replace the nonlinear signal  $u_d(t)$  by sine half-waves, too. Because the duration of the positive half-wave of  $u_d(t)$  is  $T_1$ , the positive half-wave contributes the portion

$$\frac{2}{\pi} K_d \frac{T_1}{T_1 + T_2}$$

to  $\overline{u_d}$ , while the negative half-wave, whose duration is  $T_2$ , contributes the portion

$$-\frac{2}{\pi} K_d \frac{T_2}{T_1 + T_2}$$

Hence, the average signal  $\overline{u_d}$  is given by

$$\overline{u_d} = \frac{2}{\pi} K_d \frac{T_1 - T_2}{T_1 + T_2} \quad (\text{A.8})$$

To get a simple expression for  $\overline{u_d}$ ,  $T_1$  and  $T_2$  are expanded to a Taylor series and terms of second and higher order are dropped. Then, we get for  $T_1$  and  $T_2$ :

$$T_1 = \frac{\pi}{\Delta\omega \left[ 1 - \frac{2}{\pi} \frac{F_H K_d K_0}{\Delta\omega N} \right]} \approx \frac{\pi}{\Delta\omega} \left( 1 + \frac{2}{\pi} \frac{F_H K_d K_0}{\Delta\omega N} \right) \quad (\text{A.9a})$$

$$T_2 = \frac{\pi}{\Delta\omega \left[ 1 + \frac{2}{\pi} \frac{F_H K_d K_0}{\Delta\omega N} \right]} \approx \frac{\pi}{\Delta\omega} \left( 1 - \frac{2}{\pi} \frac{F_H K_d K_0}{\Delta\omega N} \right) \quad (\text{A.9b})$$

With these approximations,  $\overline{u_d}$  becomes

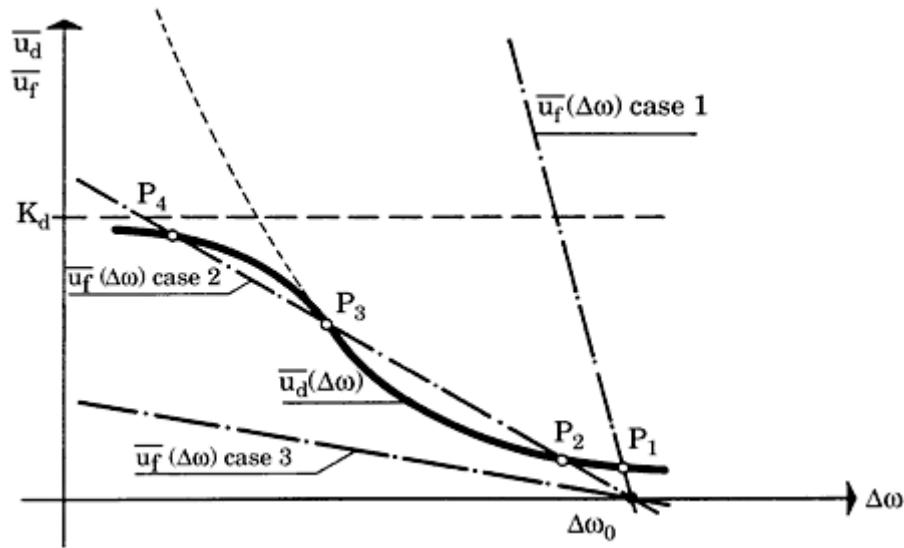
$$\overline{u_d} = \frac{4 F_H K_d^2 K_0}{\pi^2 \Delta\omega N} \quad (\text{A.10a})$$

This can be written in a simplified form as

$$\overline{u_d} = \frac{c_d}{\Delta\omega} \quad (\text{A.10b})$$

with

$$c_d = \frac{4F_H K_d^2 K_0}{\pi^2 N}$$



**Figure A.3** A plot of the average signals  $\bar{u}_d(t)$  and  $\bar{u}_f(t)$  against the frequency offset  $\Delta\omega$  for the unlocked PLL.

Thus, the average phase-detector output signal  $\bar{u}_d$  is inversely proportional to  $\Delta\omega$ . This is valid for large values of  $\Delta\omega$  only, because  $\bar{u}_d$  can never become larger than  $K_d$ , as is easily seen from Fig. A.2. If the average  $u_d$  signal is plotted against  $\Delta\omega$  (see Fig. A.3), the curve is a hyperbola for large values of  $\Delta\omega$  but approaches  $K_d$  for small  $\Delta\omega$ .

Next, we derive an expression for the average loop filter output signal  $\bar{u}_f$  as a function of the frequency offset  $\Delta\omega$ . As defined by Eq. (1.1), the (down-scaled) average frequency of  $\omega_{20}'$  of the VCO is given by

$$\bar{\omega}_{20}' = \omega_0' + \frac{K_0}{N} \bar{u}_f \quad (\text{A.11})$$

where  $\bar{u}_f$  is the average output signal of the loop filter. With the abbreviations  $\Delta\omega = \omega_1 - \omega_{20}'$  and  $\Delta\omega_0 = \omega_1 - \omega_0'$ , this can be written as

$$\Delta\omega = \Delta\omega_0 - \frac{K_0}{N} \bar{u}_f$$

or

$$\bar{u}_f = \frac{\Delta\omega_0 - \Delta\omega}{K_0/N} \quad (\text{A.12})$$

Plotting  $\bar{u}_f$  against  $\Delta\omega$  yields a straight line. This line is also shown in Fig. A.3. From this illustration, we can determine whether or not a pull-in process will take place. Depending on

the slope of  $\overline{u_f}$  against  $\Delta\omega$ , the line  $\overline{u_f}(\Delta\omega)$  can intersect with the curve  $\overline{u_d}(\Delta\omega)$  at one point (case 1), at three points (case 2), or at no point at all (case 3). Consider case 1 first. The curves  $\overline{u_d}(\Delta\omega)$  and  $\overline{u_f}(\Delta\omega)$  intersect at point  $P_1$ . In this case, the frequency of the VCO is pulled up slightly, but the system remains “hung” in point  $P_1$ . An analysis of stability shows that point  $P_1$  is a stable point; thus, no pull-in process will occur.

In case 2, the curves  $\bar{u}_d(\Delta\omega)$  and  $\bar{u}_f(\Delta\omega)$  intersect at points  $P_2$ ,  $P_3$ , and  $P_4$ . A stability analysis shows that points  $P_2$  and  $P_4$  are stable, but  $P_3$  is unstable. After power-on, the system will thus remain hung at point  $P_2$ .

Apparently, a pull-in process takes place only when there is no point of intersection, as in case 3. The two curves do not intersect if the equation

$$\bar{u}_f(\Delta\omega) = \bar{u}_d(\Delta\omega) \quad (\text{A.13})$$

does not have a real solution.

Inserting Eqs. (A.10b) and (A.12) into (A.13) yields the quadratic equation

$$\Delta\omega^2 - \Delta\omega_0 \Delta\omega + \frac{K_0}{N} c_d = 0 \quad (\text{A.14})$$

Its solution is

$$\Delta\omega_{1,2} = \frac{\Delta\omega_0 \pm \sqrt{\Delta\omega_0^2 - 4\frac{K_0}{N}c_d}}{2\frac{K_0}{N}c_d} \quad (\text{A.15})$$

The roots become complex if the discriminant (the expression under the radical) becomes negative. Putting the discriminant to zero yields the limiting case  $\Delta\omega_0 = \Delta\omega_P$ ; that is, the pull-in range  $\Delta\omega_P$  is the initial frequency offset  $\Delta\omega_0$  for which the discriminant becomes zero. If we perform this calculation, we obtain

$$\Delta\omega_P = 2\sqrt{\frac{K_0}{N}c_d} \quad (\text{A.16})$$

If we insert  $c_d$  from Equation (A.10a) and make use of the substitutions defined in Eqs. (3.13) and (3.14), we finally get the expressions

$$\Delta\omega_P = \frac{4}{\pi} \sqrt{2\zeta\omega_n \frac{K_0}{N} K_d - \omega_n^2} \quad (\text{A.17a})$$

for a passive lead-lag filter and a low-gain loop,

$$\Delta\omega_P = \frac{4\sqrt{2}}{\pi} \sqrt{\zeta\omega_n \frac{K_0}{N} K_d} \quad (\text{A.17b})$$

for a passive lead-lag filter and a high-gain loop,

$$\Delta\omega_P = \frac{4}{\pi} \sqrt{2\zeta\omega_n \frac{K_0}{N} K_d - \frac{\omega_n^2}{K_a}} \quad (\text{A.18a})$$

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).

Copyright ©2004 The McGraw-Hill Companies. All rights reserved.

Any use is subject to the Terms of Use as given at the website.

for an active lead-lag filter and a low-gain loop,

$$\Delta\omega_P = \frac{4\sqrt{2}}{\pi} \sqrt{\zeta\omega_n \frac{K_0}{N} K_d} \quad (\text{A.18b})$$

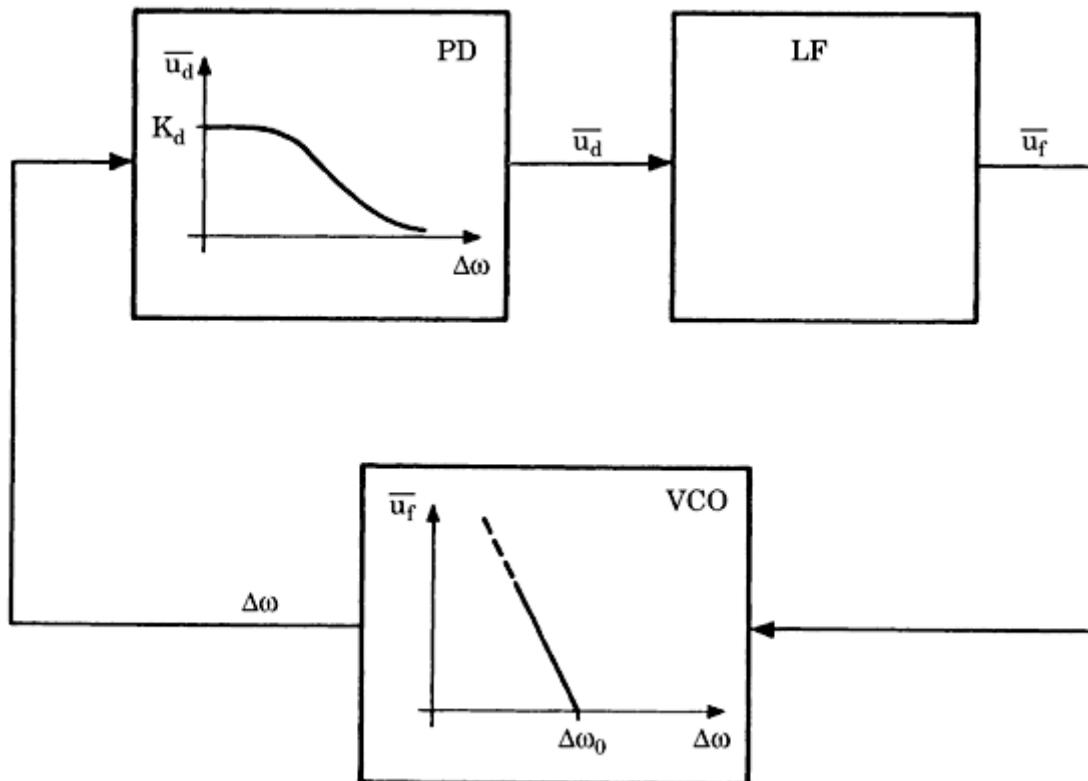
for an active lead-lag filter and a high-gain loop.

Because most loops are high-gain loops, Eq. (A.17b) can be used for most active and passive lead-lag filters. The situation is completely different if the active PI loop filter is used. Since the gain at low frequencies approaches infinity for this type of filter, the pull-in range becomes infinite, too, as explained in Sec. 3.9.3.

We conclude that the model shown in Fig. A.3 enabled us to calculate the pull-in range  $\Delta\omega_P$ . To get that result, we simply postulated that the two curves for  $\bar{u}_d(\Delta\omega)$  and  $\bar{u}_f(\Delta\omega)$  shall not intersect. In the following section, we will see that a slightly expanded model leads to an approximation for the pull-in time  $T_p$ .

## A Simplified Model for the Pull-in Time $T_p$ of the LPLL

To get an approximation for the pull-in time, we now try to find a differential equation that tells us how the instantaneous frequency offset  $\Delta\omega = \omega_1 - \omega_{20}'$  varies with time  $t$ . A model for the pull-in process is shown in Fig. A.4. To compute the pull-in process, the transfer functions of the three building blocks must



**Figure A.4** A mathematical model for the pull-in process of the PLL.

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

be known for the *unlocked state* of the PLL. The models for the phase detector and for the VCO have already been derived in the preceding section. We found that the average output signal  $\bar{u}_d$  varies inversely proportional to the frequency offset  $\Delta\omega = \omega_1 - \omega_{20}'$ , where  $\omega_{20}'$  is the down-scaled average instantaneous radian frequency  $\omega_2(t)$  of the VCO. [Equation \(A.10a\)](#) describes phase-detector performance in the unlocked state. Since the variable  $\Delta\omega$  appears in the denominator—and not in the numerator—of [Eq. \(A.10a\)](#), the differential equation yielding  $\Delta\omega(t)$  will certainly be nonlinear, which precludes the use of the Laplace transform. The pull-in process must therefore be calculated in the time domain. The operation of the VCO is governed by [Eq. \(A.12\)](#), which is linear. To complete the analysis, we must introduce the differential equation of the loop filter. If we assume for the moment that the passive lead-lag filter is used, its transfer function is given by

$$F(s) = \frac{1 + s\tau_2}{1 + s(\tau_1 + \tau_2)} \quad (\text{A.19a})$$

When transformed back into the time domain, we have

$$\bar{u}_f(t) + (\tau_1 + \tau_2) \frac{d}{dt} \bar{u}_f(t) = \bar{u}_d(t) + \tau_2 \frac{d}{dt} \bar{u}_d(t) \quad (\text{A.19b})$$

The three equations [\(A.10a\)](#), [\(A.12\)](#), and [\(A.19b\)](#) fully describe the pull-in process. When  $\bar{u}_d$  and  $\bar{u}_f$  are eliminated, we get the nonlinear differential equation

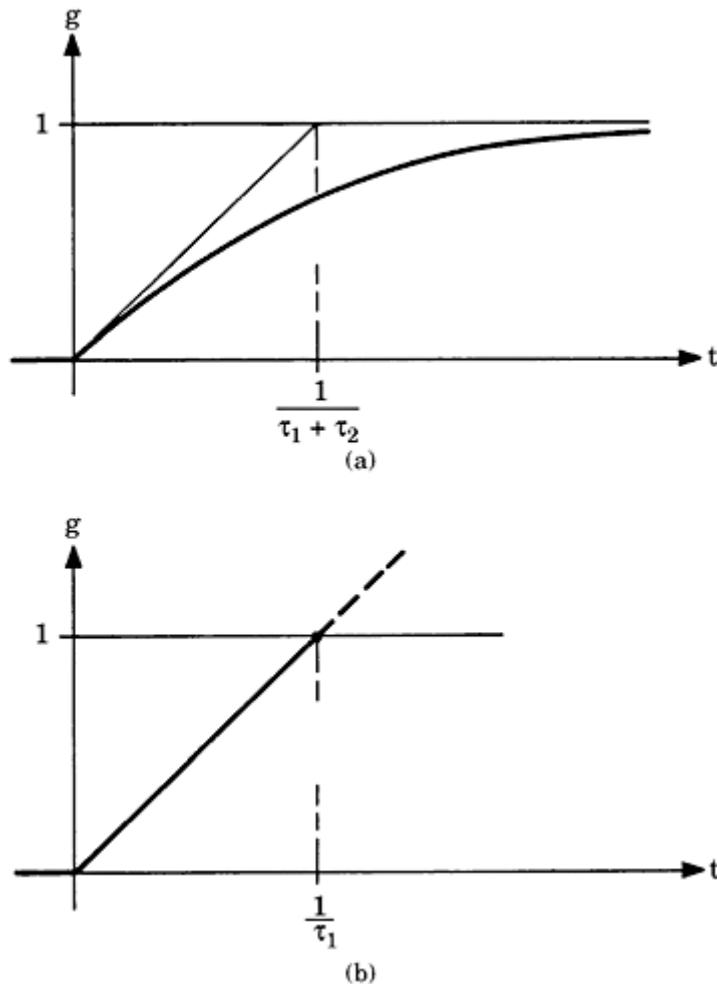
$$\frac{d}{dt} \Delta\omega \left( \frac{\tau_1 + \tau_2}{K_0/N} - \frac{\tau_2 c_d}{\Delta\omega^2} \right) + \frac{c_d}{\Delta\omega} + \frac{\Delta\omega}{K_0/N} = \frac{\Delta\omega_0}{K_0/N} \quad (\text{A.20})$$

for  $\Delta\omega$  versus  $t$ . Of course, such a differential equation can be numerically integrated on a computer, but unfortunately this does not deliver us an explicit expression for the pull-in time. In order to get such a formula, the differential equation should be simplified such that we get an algebraic expression for  $\Delta\omega$  versus time. The solution becomes much simpler if we approximate the transfer function of the loop filter by

$$F(s) = \frac{1}{s\tau_1} \quad (\text{A.21a})$$

which is the transfer function of an ideal integrator. The following consideration shows under which circumstances this approximation is acceptable. [Figure A.5a](#) shows the step response  $g(t)$  of a passive lead-lag filter having the transfer function given in [Eq. \(A.19a\)](#) for the case  $\tau_1 \gg \tau_2$ . In [Fig. A.5b](#), the step response  $g(t)$  of the ideal integrator [[Eq. \(A.21a\)](#)] is shown. The initial slope of both step responses is the same. It follows that the ideal integrator is a valid approximation to the passive lag filter as long as events are considered whose duration is not much more than the loop filter time constant  $\tau_1$ . Experience shows that this is not necessarily fulfilled. In cases where  $T_p$  turns out to be less than  $\tau_1$ ,

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.



**Figure A.5** The step response of first-order low-pass filters. (a) A step response of a passive lag filter. (b) A step response of the ideal integrator.

the approximation is quite good—in other words, the errors are normally below 10 percent of the predicted value. When  $T_p$  shows up to be larger than  $\tau_1$ , however, the approximation becomes rather crude. Finally, if the initial frequency offset  $\Delta\omega_0$  comes close to the pull-in range  $\Delta\omega_p$ , the pull-in time approaches infinity. As a rule of thumb, the formula for  $T_p$ , which will be derived in the following, delivers reasonable estimates (errors < 10 percent) when  $\Delta\omega_0$  is less than about 0.8  $\Delta\omega_p$ .

Transforming [Eq. \(A.21a\)](#) back into time domain yields

$$\bar{u}_d(t) = \tau_1 \frac{d}{dt} \bar{u}_f(t) \quad (\text{A.21b})$$

Inserting Eqs. [\(A.10a\)](#) and [\(A.21b\)](#) into [Eq. \(A.12\)](#) yields the simplified differential equation

$$\frac{d}{dt} \Delta\omega \frac{\tau_1}{K_0/N} + \frac{c_d}{\Delta\omega} = 0 \quad (\text{A.22})$$

Although this equation is still nonlinear, the variables  $\Delta\omega$  and  $t$  can be separated, so we have

$$\frac{\tau_1}{c_d K_0 / N} \int_{\Delta\omega_0}^{\Delta\omega_L} \Delta\omega d\Delta\omega = - \int_0^{T_p} dt \quad (\text{A.23})$$

The limits of integration on the left side are  $\Delta\omega_0$  and  $\Delta\omega_L$ , which says that the pull-in process ends when the instantaneous frequency offset  $\Delta\omega$  falls below the lock range  $\Delta\omega_L$ . At that time, an ordinary lock-in process follows. The limits of integration on the right side are 0 and  $T_p$ . In most cases,  $\Delta\omega_0$  is much larger than  $\Delta\omega_L$ , so we finally get

$$T_p \approx \frac{\tau_1 \Delta\omega_0^2}{2 c_d K_0 / N} \quad (\text{A.24})$$

Inserting  $c_d$  from [Equation \(A.10a\)](#) and using the substitutions of [Eq. \(3.13\)](#), we get for the pull-in time

$$T_p \approx \frac{\pi^2}{16} \frac{\Delta\omega_0^2}{\zeta \omega_n^3} \quad (\text{A.25a})$$

If the loop filter is an active lead-lag filter, its transfer function can also be replaced by the transfer function of the ideal integrator. When the computation is repeated for the active lead-lag filter, the pull-in time [using the substitutions of [Eq. \(3.14\)](#)] becomes

$$T_p \approx \frac{\pi^2}{16} \frac{\Delta\omega_0^2 K_a}{\zeta \omega_n^3} \quad (\text{A.25b})$$

Finally, when the loop filter is an active PI filter, the pull-in time [using the substitutions of [Eq. \(3.15\)](#)] is given by

$$T_p \approx \frac{\pi^2}{16} \frac{\Delta\omega_0^2}{\zeta \omega_n^3} \quad (\text{A.25c})$$

## The Pull-in Range $\Delta\omega_p$ of the DPLL

The approach described in [Sec. A.1](#) can be adopted to also calculate the pull-in range of the digital PLL. We assume first that the EXOR gate is used as the phase detector. As explained in [Sec. 3.9.3](#) and in [Fig. 3.16](#), the output signal of the PD in the unlocked state is no longer sinusoidal but is an asymmetrical triangular wave. For a given average frequency offset  $\Delta\omega$ , we can again calculate the resulting DC component of the phase-detector output signal  $\bar{u}_d$ . The result is

$$\overline{u_d} = \frac{\pi^2 K_d^2 K_0 F_H}{16 \Delta\omega N} \quad (\text{A.26a})$$

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

which can be rewritten as

$$\bar{u_d} = \frac{c_d}{\Delta\omega} \quad (\text{A.26b})$$

with

$$c_d = \frac{\pi^2 K_d^2 K_0 F_H}{16 N}$$

The characteristics of the loop filter and of the VCO stay the same as in the case of the LPPLL, so the pull-in range is obtained simply by inserting the  $c_d$  into Eq. (A.16). The result then becomes

$$\Delta\omega_P = \frac{\pi}{2} \sqrt{2\zeta\omega_n K_0 K_d / N - \omega_n^2} \quad (\text{A.27a})$$

(PD = EXOR, passive lead-lag filter, low-gain loop)

$$\Delta\omega_P = \frac{\pi}{\sqrt{2}} \sqrt{\zeta\omega_n K_0 K_d / N} \quad (\text{A.27b})$$

(PD = EXOR, passive lead-lag filter, high-gain loop)

$$\Delta\omega_P = \frac{\pi}{2} \sqrt{2\zeta\omega_n K_0 K_d / N - \frac{\omega_n^2}{K_a}} \quad (\text{A.28a})$$

(PD = EXOR, active lead-lag filter, low-gain loop)

$$\Delta\omega_P = \frac{\pi}{\sqrt{2}} \sqrt{\zeta\omega_n K_0 K_d / N} \quad (\text{A.28b})$$

(PD = EXOR, active lead-lag filter, high-gain loop).

The pull-in range for a DPPLL using a JK-flipflop phase detector remains to be calculated. As shown in Sec. 3.9.3 and Fig. 3.17, the phase-detector output signal is an unsymmetrical sawtooth in the unlocked state. Again using the same argumentation as in Sec. A.1, the coefficient  $c_d$  can be shown to be

$$c_d = \frac{\pi^2 K_d^2 K_0 F_H}{4 N}$$

in this case. If we insert  $c_d$  into Eq. (A.16) and make use of the substitutions in Eqs. (3.13) and (3.14), we get

$$\Delta\omega_P = \pi \sqrt{2\zeta\omega_n K_0 K_d / N - \omega_n^2}$$

(A.29a)

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

(PD = JK-flipflop, passive lead-lag filter, low-gain loop)

$$\Delta\omega_P = \pi \sqrt{2\zeta\omega_n K_0 K_d / N} \quad (\text{A.29b})$$

(PD = JK-flipflop, passive lead-lag filter, high-gain loop)

$$\Delta\omega_P = \pi \sqrt{2\zeta\omega_n K_0 K_d / N - \frac{\omega_n^2}{K_a}} \quad (\text{A.30a})$$

(PD = JK-flipflop, active lead-lag filter, low-gain loop)

$$\Delta\omega_P = \pi \sqrt{2\zeta\omega_n K_0 K_d / N} \quad (\text{A.30b})$$

(PD = JK-flipflop, active lead-lag filter, high-gain loop).

## The Pull-in Time $T_p$ of the DPLL

As we have seen in [Sec. A.3](#), the behavior of LPLLs and DPLLs in the unlocked state is very similar. For all types of phase detectors used, the average output signal can be represented by the same formula

$$\bar{u}_d = \frac{c_d}{\Delta\omega}$$

where only the coefficient  $c_d$  depends on the phase-detector type. Therefore, the same differential equation is valid for the pull-in process of DPLLs. The pull-in time  $T_p$  is found by inserting the appropriate expression for  $c_d$  into [Eq. \(A.24\)](#). For the DPLL using the EXOR phase detector, the pull-in time  $T_p$  becomes [making use of the substitutions of Eqs. [\(3.13\)](#) to [\(3.15\)](#)]

$$T_p \approx \frac{4}{\pi^2} \frac{\Delta\omega_0^2}{\zeta\omega_n^3} \quad (\text{A.31a})$$

in the case of the passive lead-lag filter,

$$T_p \approx \frac{4}{\pi^2} \frac{\Delta\omega_0^2 K_a}{\zeta\omega_n^3} \quad (\text{A.31b})$$

in the case of the active lead-lag filter, and

$$(\text{A.31c})$$

$$T_P \approx \frac{4}{\pi^2} \frac{\Delta\omega_0^2}{\zeta\omega_n^3}$$

in the case of the active PI filter.

When the DPLL uses the JK-flipflop phase detector, the pull-in time  $T_P$  becomes [making use of the substitutions in Eqs. (3.13) to (3.15)]

$$T_P \approx \frac{1}{\pi^2} \frac{\Delta\omega_0^2}{\zeta\omega_n^3} \quad (\text{A.32a})$$

in the case of the passive lead-lag filter,

$$T_P \approx \frac{1}{\pi^2} \frac{\Delta\omega_0^2 K_a}{\zeta\omega_n^3} \quad (\text{A.32b})$$

in the case of the active lead-lag filter, and

$$T_P \approx \frac{1}{\pi^2} \frac{\Delta\omega_0^2}{\zeta\omega_n^3} \quad (\text{A.32c})$$

in the case of the active PI filter.

As we stated in [Sec. 3.9.3](#), the approximations for the pull-in time are valid if the initial frequency offset  $\Delta\omega_0$  is markedly below the pull-in range  $\Delta\omega_p$ —in other words, below about 0.8  $\Delta\omega_p$ . If the initial frequency offset comes close to  $\Delta\omega_p$ , the pull-in time approaches infinity.

# The Laplace Transform

## Transforms Are the Engineer's Tools

Trying to solve electronic problems without using the Laplace transform is similar to traveling through a foreign country with a globe instead of a map ([Fig. B.1](#)). An engineer who tries to find the transient response of an electric network to an impulse function by solving differential equations, for example, certainly is working with inadequate tools (see [Fig. B.2](#)). But the engineer familiar with the techniques of the Laplace transform may find a solution very quickly, as shown in [Fig. B.3](#).

A map images a three-dimensional object to a plane. Every spatial point of the three-dimensional object is represented by a unique point in the plane of the map. Things are similar, though different, in the case of the Laplace transform. Here, a function in the *time domain* (such as an electric signal) is transformed to another function in the *complex frequency domain*. The trouble with the Laplace transform starts right here: even an electronic hobbyist can imagine what the frequency spectrum of an electric signal is, but what is *complex frequency*?

One need not be a cow to know what milk is, but it is surely easier to understand the term complex frequency if we first consider *real* frequency spectra. The Laplace transform is a more general form of the Fourier transform; in other words, the Fourier transform is a special case of the Laplace transform. In this context, it may be easier to start with the special case before proceeding to the more general one.

The Fourier transform is explained best by first looking at a periodic signal such as a square wave ([Fig. B.4](#)). The angular frequency of this signal is assumed to be  $\omega_0$ . This square wave can now be thought to be composed of an infinite number of sine-wave signals having frequencies  $\omega_0$  (fundamental frequency),  $2\omega_0$ ,  $3\omega_0$ , and so on (harmonics). Mathematically, any periodic signal  $f(t)$  having a repetition frequency of  $\omega_0$  can be written as a sum of its harmonics:

$$f(t) = \sum_{n=-\infty}^{+\infty} F(n\omega_0) \exp(jn\omega_0 t) \quad (\text{B.1})$$



**Figure B.1** Trying to solve electronic problems without using the Laplace transform is as cumbersome as traveling through a foreign country with a globe instead of a map.

The so-called Fourier coefficients  $F(n\omega_0)$  are calculated from

$$F(n\omega_0) = \frac{1}{T} \int_{-T/2}^{T/2} f(t) \exp(-jn\omega_0 t) dt \quad (\text{B.2})$$

where  $T$  is the period of the periodic signal  $f(t)$ ,  $T = 2\pi/\omega_0$ , and  $F(n\omega_0)$  is the amplitude of the harmonic component with frequency  $n\omega_0$ . Note that the Fourier coefficients  $F(n\omega_0)$  are generally complex numbers. We can therefore write

$$F(n\omega_0) = |F(n\omega_0)| \exp(j\Phi_n) \quad (\text{B.3})$$

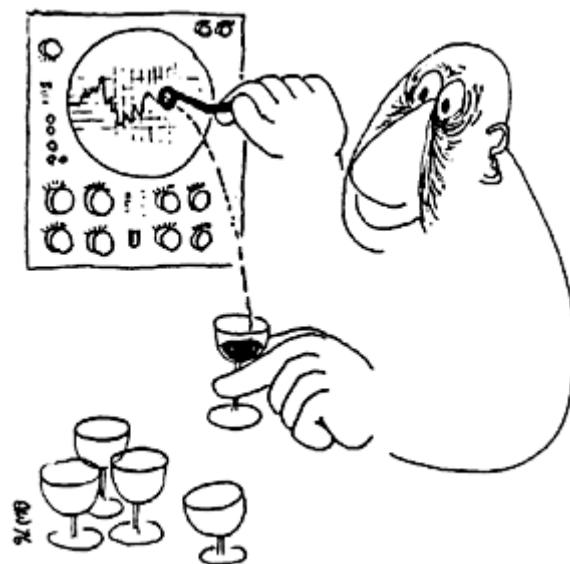
where  $|F(n\omega_0)|$  is the amplitude and  $\Phi_n$  is the phase of  $F(n\omega_0)$ .



**Figure B.2** Looking at the transient response of electric networks without using the Laplace transform can be tricky ...

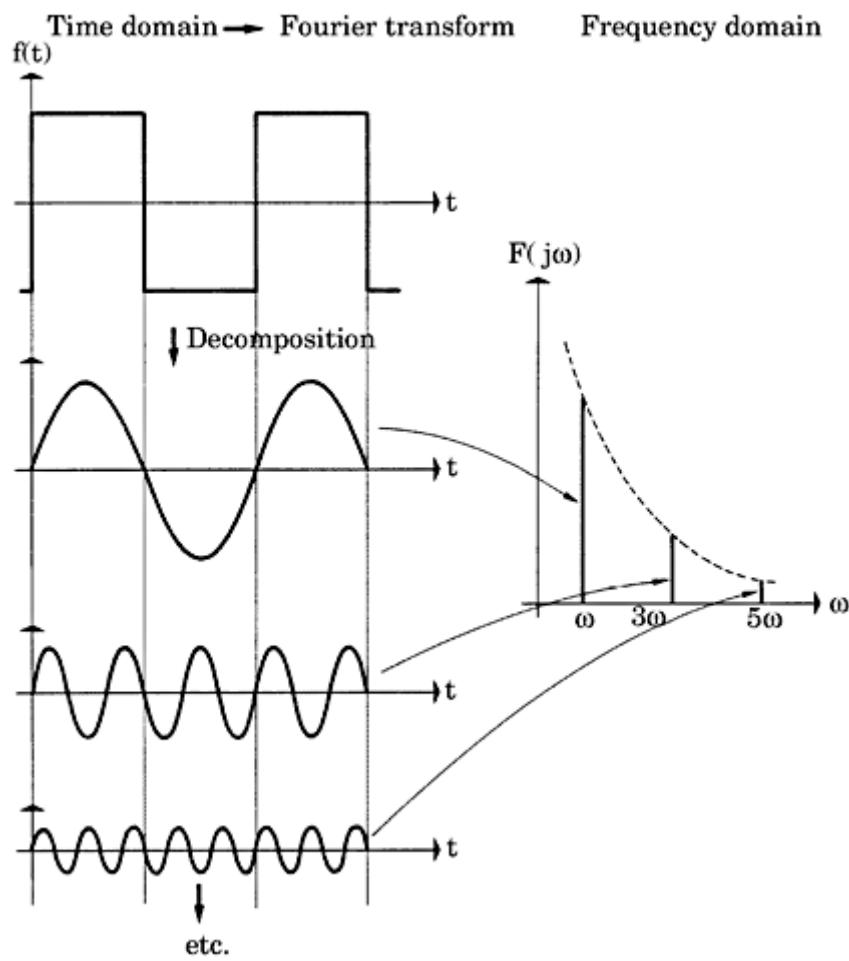
---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.



**Figure B.3** ... but the engineer familiar with Laplace techniques may find the solution very quickly.

When plotting the Fourier transform of a signal  $f(t)$ , the amplitude  $|F(n\omega_0)|$  and the phase  $\Phi_n$  are normally plotted against  $\omega$ ; these functions are called *amplitude* and *phase spectra*, respectively. In the case of periodic functions  $f(t)$ , the Fourier spectra become discrete—that is, the Fourier coefficients  $F(n\omega_0)$  are defined only at the discrete frequencies  $\omega_0, 2\omega_0, 3\omega_0, \dots$



**Figure B.4** The principle of the Fourier transform.

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
 Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
 Any use is subject to the Terms of Use as given at the website.

[Figure B.4](#) shows the amplitude spectrum  $|F(n\omega_0)|$  of a symmetrical square wave; for a symmetrical waveform it can be shown that the even harmonics disappear. Consequently, the Fourier spectrum of [Fig. B.4](#) only shows lines at  $\omega_0$ ,  $3\omega_0$ ,  $5\omega_0$ , and so on.

In real life, we find many signals that are not periodic, such as a single pulse. What about the Fourier transform of such signals? If a signal  $f(t)$  is not periodic, we could state that its period approaches infinity, which means its fundamental frequency approaches zero. If the fundamental frequency approaches zero, the spectral lines in [Eq. \(B.1\)](#) come closer and closer, and finally the sum is replaced by an integral,

$$f(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega) e^{j\omega t} dt \quad (\text{B.4})$$

For an aperiodic signal  $f(t)$ , the Fourier spectrum  $F(\omega)$  becomes continuous;  $F(\omega)$  is called the *Fourier transform* of the signal  $f(t)$ . The Fourier transform is calculated in the same way as the Fourier coefficients  $F(n\omega_0)$  in [Eq. \(B.2\)](#). If  $T$  approaches  $\infty$  in this equation, we get

$$F(\omega) = \int_{-\infty}^{\infty} f(t) e^{-j\omega t} dt \quad (\text{B.5})$$

This is the definition of the *continuous Fourier transform*. Note that the Fourier transform  $F(\omega)$  for any given signal  $f(t)$  can be calculated by evaluating the integral in [Eq. \(B.5\)](#). If the Fourier transform of a signal  $f(t)$  is given first, the corresponding signal  $f(t)$  in the time domain can be obtained by applying [Eq. \(B.4\)](#). This equation is also called the *inverse Fourier transform*. The Fourier transforms  $F(\omega)$  for the most important signal waveforms have been tabulated in many reference books.<sup>36</sup>

The usefulness of the Fourier transform is limited by a serious drawback: If we try to evaluate the Fourier integral in [Eq. \(B.5\)](#), we find that the integral does *not converge* for most signals  $f(t)$ . You will find it impossible to find a solution of the Fourier integral by conventional methods, even for such a simple signal as  $f(t) = \sin\omega_0 t$ . The Laplace transform offers a way to circumvent this problem.

## Laplace Transform Is the Key to Success

Imagine we would like to know the Fourier transform of an extremely simple signal,  $f(t) = \sin\omega_0 t$  [with  $f(t) = 0$  for  $t < 0$ ]. Using the definition of the Fourier transform, [Eq. \(B.5\)](#), we get

$$F(\omega) = \int_0^{\infty} f(t) e^{-j\omega t} dt = \int_0^{\infty} \sin\omega_0 t e^{-j\omega t} dt$$

**Any use is subject to the Terms of Use as given at the website.**

Using an integral table we find for  $F(\omega)$

$$F(\omega) = \frac{-j\omega e^{-j\omega t} \sin \omega_0 t - \omega_0 e^{-j\omega t} \cos \omega_0 t}{\omega_0^2 - \omega^2} \Big|_0^\infty$$

Introducing the limits of integration (here 0 and  $\infty$ ) yields

$$F(\omega) = \frac{-j\omega e^{-j\infty} \sin \infty - \omega_0 e^{-j\infty} \cos \infty + \omega_0}{\omega_0^2 - \omega^2}$$

But what is  $\sin \infty$ , and what is  $\cos \infty$ ? Both functions are periodic and are within a range of  $-1$  to  $1$ ; hence  $\sin \infty$  and  $\cos \infty$  are not defined.

The evaluation of the Fourier integral would be simpler if the function  $f(t)$  would approach zero for very large values of  $t$ . The solution of the Fourier integral becomes possible if  $f(t)$  is multiplied by a damping function  $e^{-\sigma t}$ , where  $\sigma$  is a positive real number:

$$F'(\omega, \sigma) = \int_0^\infty [f(t)e^{-\sigma t}]e^{-j\omega t} dt \quad (\text{B.6})$$

The notation  $F'$  ( $\omega, \sigma$ ) is chosen to emphasize that now  $F'$  is also dependent on  $\sigma$ . The prime is furthermore chosen to differentiate  $F'$  from the Fourier integral in [Eq. \(B.5\)](#). The product  $f(t)e^{-\sigma t}$  approaches zero for large  $t$ . This is true even when  $f(t)$  is an exponential function  $f(t) = e^{at}$  with positive  $a$ . If  $\sigma$  is chosen larger than  $a$ , the product approaches zero for  $t \rightarrow \infty$ . The modified Fourier integral in [Eq. \(B.6\)](#) is now easily evaluated. Let us perform the calculation for the previous example  $f(t) = \sin \omega_0 t$ . We then have

$$F'(\omega, \sigma) = \int_0^\infty \sin \omega_0 t e^{-\sigma t} e^{-j\omega t} dt$$

Again using an integral table, we get

$$F'(\omega, \sigma) = \frac{-(\sigma + j\omega)e^{-(\sigma + j\omega)t} \sin \omega_0 t - \omega_0 e^{-(\sigma + j\omega)t} \cos \omega_0 t}{\omega_0^2 + (\sigma + j\omega)^2} \Big|_0^\infty$$

If the limits of integration are now inserted, the term

$$e^{-(\sigma + j\omega)\infty}$$

becomes zero for positive  $\sigma$ . Then  $F'$  ( $\omega, \sigma$ ) becomes

$$F'(\omega, \sigma) = \frac{\omega_0}{\omega_0^2 + (\sigma + j\omega)^2}$$

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

Of course,  $F'(\omega, \sigma)$  contains the damping factor  $\sigma$ . The Fourier integral  $F(\omega)$  is now obtained simply by letting  $\sigma \rightarrow 0$ :

$$F(\omega) = \lim_{\sigma \rightarrow 0} F'(\omega, \sigma) = \frac{\omega_0}{\omega_0^2 - \omega^2}$$

If  $\sigma$  is set equal to zero in [Eq. \(B.6\)](#), this equation is transformed into [Eq. \(B.5\)](#). Thus, the Fourier transform of any function  $f(t)$  is obtained by first introducing a damping function  $e^{-\sigma t}$ , evaluating the integral [[Eq. \(B.6\)](#)] for  $\sigma > 0$  and finally letting  $\sigma \rightarrow 0$ .

Let's now see what the Laplace transform really is. [Equation. \(B.6\)](#) can also be written in a different form:

$$F'(\omega, \sigma) = \int_0^\infty f(t) [e^{-\sigma t} e^{-j\omega t}] dt$$

In contrast to [Eq. \(B.6\)](#), we now combine the damping function  $e^{-\sigma t}$  with the exponential function  $e^{-j\omega t}$ . This can be written as

$$F'(\omega, \sigma) = \int_0^\infty f(t) e^{-(\sigma+j\omega)t} dt$$

We now define

$$s = \sigma + j\omega$$

and call the new variable  $s$  complex frequency. Because the two variables  $\sigma$  and  $\omega$  appear only in the form  $\sigma + j\omega$ ,  $F(\omega, \sigma)$  can be written as  $F(\sigma + j\omega) = F(s)$ . We then have

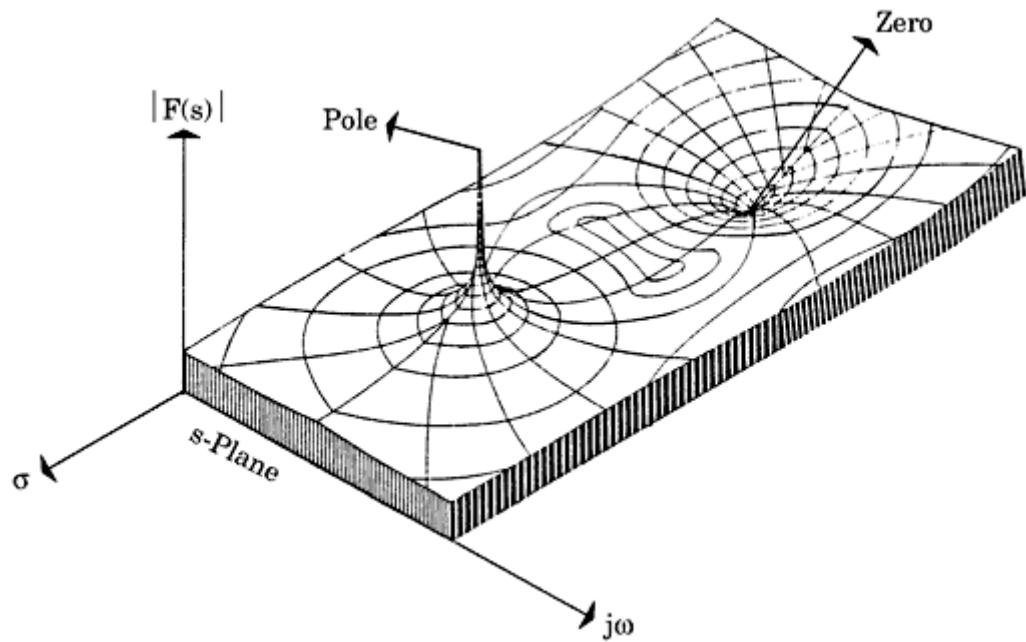
$$F(s) = \int_0^\infty f(t) e^{-st} dt \tag{B.7}$$

This is the definition of the Laplace transform.

In the case of the Fourier transform,  $F(\omega)$  was a *complex* function of the *real* variable  $\omega$ . To plot  $F(\omega)$  against  $\omega$  we had to plot amplitude  $|F(\omega)|$  and phase  $\Phi(\omega)$  as a function of  $\omega$ . (This plot is commonly called a Bode diagram.) In the case of the Laplace transform, however,  $F(s)$  is a *complex* function of the *complex* variable  $s$ . Plotting  $F(s)$  is much more difficult than plotting  $F(\omega)$ . To plot  $F(s)$ , a relief of the magnitude  $|F(s)|$  and of the phase  $\Phi(s)$  could be constructed; a relief of  $|F(s)|$  is shown in [Fig. B.5](#).

The construction of such a relief is a cumbersome procedure. As we shall see later, it is sufficient to know the locations of some singular points of  $F(s)$  only, namely, the *poles* and *zeroes* of  $F(s)$ . A pole is a point in the  $s$  plane where  $F(s)$  becomes infinity, and a zero is a point in the  $s$ -plane where  $F(s)$  is zero. The so-called pole-zero plot ([Fig. B.6](#)) shows the

locations in the  $s$ -plane where  $F(s)$  has poles ( $\bullet$ ) or zeroes (0). [Note that not every function  $F(s)$  necessarily has poles]



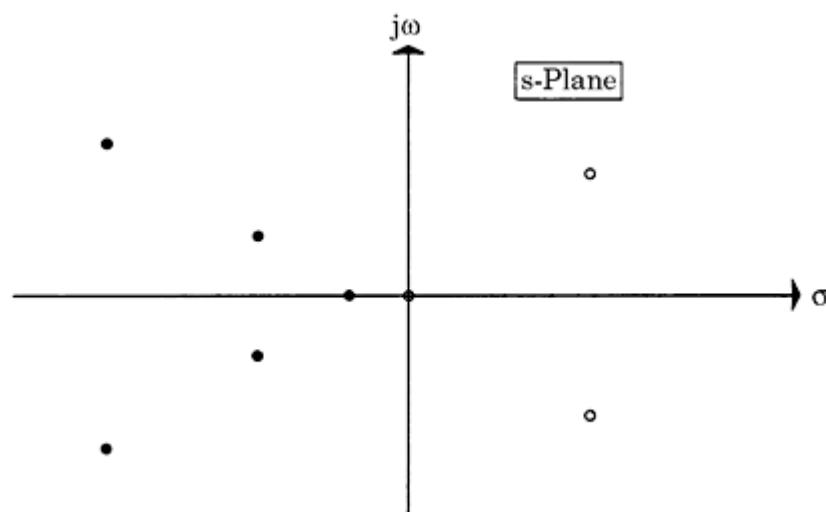
**Figure B.5** The magnitude of the Laplace transform  $F(s)$  plotted as a function of complex frequency  $s$  yields a relief. This illustration shows a pole and a zero of the transfer function  $F(s)$ .

or zeroes.] We see immediately that the Fourier transform [Eq. (B.5)] is a special case of the Laplace transform [Eq. (B.7)]. The Fourier transform  $F(\omega)$  is simply obtained by finding the Laplace transform  $F(s)$  and then putting  $\sigma = 0$

$$F(\omega) = \lim_{\sigma \rightarrow 0} F(s), \quad s = \sigma + j\omega \quad (\text{B.8})$$

In other words,  $F(\omega)$  is equal to the function  $F(s)$  on the imaginary axis of the  $s$ -plane.

Equation (B.7) shows us how to calculate the Laplace transform from a given function  $f(t)$  in the time domain. As was the case with the Fourier transform, it



**Figure B.6** The lot of the poles and zeroes of a given transfer function  $F(s)$ .

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

is also possible to calculate  $f(t)$  if  $F(s)$  is given first. This transformation is called the *inverse* Laplace transform and is defined by

$$f(t) = \frac{1}{2\pi j} \int_{c-j\infty}^{c+j\infty} F(s)e^{-st}ds \quad (\text{B.9})$$

where  $c$  is a real constant. As is seen from [Eq. \(B.9\)](#), the path of integration is a line parallel to the imaginary axis.[1,3,18](#)

In most cases, computation of the Laplace transform of the inverse Laplace transform by evaluating Eqs. [\(B.7\)](#) and [\(B.9\)](#) is no longer necessary since transformation tables are available (see [Table B.1](#)). To conclude this section, we shall introduce some convenient abbreviations:

$$F(s) = L[f(t)] \quad (\text{B.10a})$$

$$f(t) = L^{-1}[F(s)] \quad (\text{B.10b})$$

[Equation \(B.10a\)](#) says that  $F(s)$  is the Laplace transform of the signal  $f(t)$ , while [Eq. \(B.10b\)](#) states that  $f(t)$  is the inverse Laplace transform of  $F(s)$ .

## A Numerical Example of the Laplace Transform

To see how the Laplace transform  $F(s)$  of a signal  $f(t)$  could be obtained by evaluating [Eq. \(B.7\)](#), let's calculate an example.

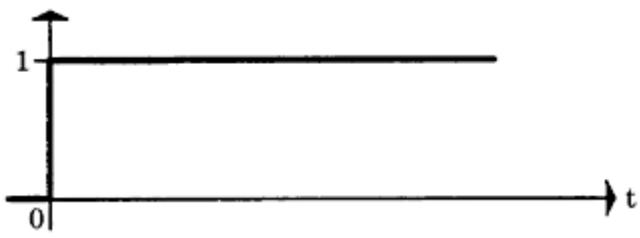
Assume  $f(t)$  is the unit step function  $f(t) = u(t)$ , as plotted in [Fig. B.7](#). For the Laplace transform  $F(s)$ , we have [according to [Eq. \(B.7\)](#)]

$$\begin{aligned} F(s) &= \int_0^{\infty} u(t)e^{-st}dt = \int_0^{\infty} e^{-st}dt = \left. \frac{e^{-st}}{-s} \right|_0^{\infty} \\ &= -\frac{1}{s}(e^{-\infty} - e^0) = -\frac{1}{s}(0 - 1) = \frac{1}{s} \end{aligned}$$

that is

$$F(s) = \frac{1}{s}$$

$F(s)$  has one single pole at  $s = 0$ —in other words, at the origin of the  $s$ -plane. Evaluating the Laplace integral in [Eq. \(B.7\)](#) for the most common test signals  $f(t)$  yields [Table B.1](#).



**Figure B.7** A plot of the unit step function against time.

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

**TABLE B.1 Laplace Transform,  $f(t) = 0$  for  $t < 0$**

	Signal $f(t)$	Laplace transform
<b>General rules</b>		
Differentiation	$\frac{df(t)}{dt}$	$sF(s) - f(0)$
Integration	$\int_0^t f(t)dt$	$\frac{F(s)}{s} + \frac{f(-1)(0)}{s}$
Time delay	$f(t - \tau)$	$F(s)e^{-s\tau}$
Multiplication by constant	$kf(t)$	$kF(s)$
Convolution	$f_1(t) \cdot f_2(t)$	$F_1(s) * F_2(s)$
<b>Functions</b>	$f_1(t) * f_2(t)$	$F_1(s) \cdot F_2(s)$
Zero	0	0
Unit step	$u(t)$	$\frac{1}{s}$
Delta function	$\delta(t)$	1
Ramp function	$t$	$\frac{1}{s^2}$
Parabola	$\frac{t^2}{2}$	$\frac{1}{s^3}$
Polynomial in t	$\frac{t^{n-1}}{(n-1)!}$	$\frac{1}{s^n}, n > 0$ (can be fraction)
<b>Exponential functions</b>	$e^{\alpha t}$	$\frac{1}{s - \alpha}$
	$\frac{1}{\alpha}(e^{\alpha t} - 1)$	$\frac{1}{s(s - \alpha)}$
	$\frac{1}{\alpha}(1 - e^{-\alpha t})$	$\frac{1}{s(s + \alpha)}$
	$\frac{t^n - 1}{(n-1)! e^{\alpha t}}$	$\frac{1}{(s - \alpha)^n}, n > 0$ (can be fraction)
<b>Trigonometric functions</b>	$\frac{1}{\alpha} \sin \alpha t$	$\frac{1}{s^2 + \alpha^2}$
	$\cos \alpha t$	$\frac{s}{s^2 + \alpha^2}$
	$\frac{1}{\alpha^2}(1 - \cos \alpha t)$	$\frac{1}{s(s^2 + \alpha^2)}$

---

**Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.**

**TABLE B.1 Laplace Transform,  $f(t) = 0$  for  $t < 0$  (Continued)**

	Signal $f(t)$	Laplace transform
Hyperbolic functions	$\frac{1}{\alpha} \sinh \alpha t$	$\frac{1}{s^2 - \alpha^2}$
	$\cosh \alpha t$	$\frac{s}{s^2 - \alpha^2}$
	$\frac{1}{\alpha^2} (\cosh \alpha t - 1)$	$\frac{1}{s(s^2 - \alpha^2)}$
Second-order systems (aperiodic)	$\frac{e^{\beta t} - e^{\alpha t}}{\beta - \alpha}$	$\frac{1}{(s - \alpha)(s - \beta)}$
	$\frac{\beta e^{\beta t} - \alpha e^{\alpha t}}{\beta - \alpha}$	$\frac{s}{(s - \alpha)(s - \beta)}$
	$\frac{\beta e^{\alpha t} - \alpha e^{\beta t}}{\alpha \beta (\alpha - \beta)} + \frac{1}{\alpha \beta}$	$\frac{1}{s(s - \alpha)(s - \beta)}$
Second-order systems (periodic)	$\frac{e^{-\zeta \omega_n t} \sin \sqrt{1 - \zeta^2} \omega_n t}{\sqrt{1 - \zeta^2} \omega_n}$	$\frac{1}{s^2 + 2s\zeta\omega_n}$
	$\left[ \cos \sqrt{1 - \zeta^2} \omega_n t - \frac{\zeta}{\sqrt{1 - \zeta^2}} \sin \sqrt{1 - \zeta^2} \omega_n t \right] e^{-\zeta \omega_n t}$	$\frac{s}{s^2 + 2s\zeta\omega_n}$
	$\left[ \frac{2\zeta^2 - 1}{\sqrt{1 - \zeta^2}} \sin \sqrt{1 - \zeta^2} \omega_n t - 2\zeta \cos \sqrt{1 - \zeta^2} \omega_n t \right] \omega_n e^{-\zeta \omega_n t}$	$\frac{s^2}{s^2 + 2s\zeta\omega_n}$
Third-order systems	$\frac{1}{\omega_n^2} \left[ 1 + (\cos \sqrt{1 - \zeta^2} \omega_n t + \frac{\zeta}{1 - \zeta^2} \sin \sqrt{1 - \zeta^2} \omega_n t) e^{-\zeta \omega_n t} \right]$	$\frac{1}{s(s^2 + 2s\zeta\omega_n)}$
	$\frac{e^{\alpha t} - [1 + (\alpha - \beta)t]e^{\beta t}}{(\alpha - \beta)^2}$	$\frac{1}{(s - \alpha)(s - \beta)}$
	$\frac{\alpha e^{\alpha t} - [\alpha + \beta(\alpha - \beta)t]e^{\beta t}}{(\alpha - \beta)^2}$	$\frac{s}{(s - \alpha)(s - \beta)}$
Various functions	$\frac{\alpha^2 e^{\alpha t} - [2\alpha - \beta + \beta(\alpha - \beta)t]\beta e^{\beta t}}{(\alpha - \beta)^2}$	$\frac{s^2}{(s - \alpha)(s - \beta)}$
	$\frac{(\beta - \gamma)e^{\alpha t} + (\gamma - \alpha)e^{\beta t} + (\alpha - \beta)e^{\gamma t}}{(\alpha - \beta)(\beta - \gamma)(\gamma - \alpha)}$	$\frac{1}{(s - \alpha)(s - \beta)(s - \gamma)}$
	$\frac{\alpha \sin \beta t - \beta \sin \alpha t}{\alpha \beta (\alpha^2 - \beta^2)}$	$\frac{1}{(s^2 + \alpha^2)(s^2 + \beta^2)}$
	$\frac{\cos \beta t - \cos \alpha t}{\alpha^2 - \beta^2}$	$\frac{s}{(s^2 + \alpha^2)(s^2 + \beta^2)}$

---

**Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.**

**TABLE B.1 Laplace Transform,  $f(t) = 0$  for  $t < 0$  (Continued)**

	Signal $f(t)$	Laplace transform
	$\frac{1}{\sqrt{\pi t}}$	$\frac{1}{\sqrt{s}}$
	$2\sqrt{\frac{t}{\pi}}$	$\frac{1}{s\sqrt{s}}$
	$\frac{n!}{(2n)!} \frac{4^n}{\sqrt{\pi}} t^{n-1/2}$	$\frac{1}{s^n\sqrt{s}}$
	$\frac{1}{\sqrt{\pi t}} e^{\alpha t}$	$\frac{1}{\sqrt{s-\alpha}}$
Error functions	$\frac{2}{\sqrt{\alpha\pi}} \int_0^{\sqrt{\alpha t}} e^{-\zeta^2} d\zeta$	$\frac{1}{s\sqrt{s+\alpha}}$
	$\frac{2e^{-\alpha t}}{\sqrt{\pi(\beta-\alpha)}} \int_0^{\sqrt{(\beta-\alpha)t}} e^{-\zeta^2} d\zeta$	$\frac{1}{(s+\alpha)\sqrt{s}}$
	$\frac{e^{-\alpha t}}{\sqrt{\pi t}} + 2\sqrt{\frac{\alpha}{\pi}} \int_0^{\sqrt{\alpha t}} e^{-\zeta^2} d\zeta$	$\frac{\sqrt{s+\alpha}}{s}$
Bessel function of zero order	$I_0(\alpha t)$	$\frac{1}{\sqrt{s^2+\alpha^2}}$
Modified Bessel function of order zero	$J_0(\alpha t)$	$\frac{1}{\sqrt{s^2-\alpha^2}}$

## Some Basic Properties of the Laplace Transform

For practical applications, it is useful to be familiar with some basic properties of the Laplace transform. The most important ones are discussed in the following subsections.

### Addition theorem

The Laplace transform, as defined by Eq. (B.7), is linear with respect to  $f(t)$ . If two signals  $f_1(t)$  and  $f_2(t)$  are given, the Laplace transform of the sum of  $f_1 + f_2$  is therefore equal to the sum

of the individual Laplace transforms  $F_1(s)$  and  $F_2(s)$ :

$$L[f_1(t) + f_2(t)] = L[f_1(t)] + L[f_2(t)] = F_1(s) + F_2(s) \quad (\text{B.11})$$

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).

Copyright ©2004 The McGraw-Hill Companies. All rights reserved.

Any use is subject to the Terms of Use as given at the website.

## Multiplication by a constant factor $k$

For the same reason we have

$$L[kf(t)] = kL[f(t)] = kF(s) \quad (\text{B.12})$$

If the signal  $f(t)$  is multiplied by a constant factor  $k$ , the Laplace transform  $F(s)$  is simply multiplied by the same factor.

## Multiplication of signals

If two signals  $f_1(t)$  and  $f_2(t)$  are multiplied together in the time domain, the Laplace transform of  $f_1(t) \cdot f_2(t)$  is *not* given by multiplying the individual Laplace transform  $F_1(s) = L[f_1(t)]$  and  $F_2(s) = L[f_2(t)]$ . Thus

$$L[f_1(t) \cdot f_2(t)] \neq F_1(s) \cdot F_2(s)$$

The operation in the complex frequency domain that corresponds to the multiplication in the time domain is much more complicated; it is called *complex convolution*.<sup>3,13,14,37</sup> We will not consider this operation in detail here, but will define it for completeness:

$$L[f_1(t) \cdot f_2(t)] = \frac{1}{2\pi j} \oint F_1(\chi) \cdot F_2(s - \chi) d\chi \quad (\text{B.13})$$

where

$$F_1(s) = L[f_1(t)]$$

$$F_2(s) = L[f_2(t)]$$

and  $\chi$  is an auxiliary complex variable. The integral on the right-hand side of Eq. (B.13) is called a *complex convolution integral*. The contour of integration is a closed loop and must be chosen on the basis of mathematical considerations<sup>3,37</sup> that will not be discussed here. For the complex convolution integral, the simplified form

$$\frac{1}{2\pi j} \oint F_1(\chi) \cdot F_2(s - \chi) d\chi = F_1(s) * F_2(s)$$

is often used.

Now we determine which operation in the time domain corresponds to a multiplication in the complex frequency domain. The result is given without a mathematical derivation:

$$(\text{B.14})$$

$$L^{-1}[F_1(s) \cdot F_2(s)] = \int_0^t f_1(\tau) f_2(t - \tau) d\tau$$

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

The integral on the right-hand side of Eq. (B.14) is called *convolution* and is often written as

$$\int_0^t f_1(\tau) f_2(t - \tau) d\tau = f_1(t) * f_2(t)$$

## Delay in the time domain

A signal  $f(t)$  and its Laplace transform  $F(s)$  are given. The signal is now delayed by the time interval  $\tau$  (see Fig. B.8). What is the Laplace transform of the delayed signal  $f(t - \tau)$ ? The derivation<sup>3,37</sup> yields

$$L[f(t - \tau)] = F(s)e^{-s\tau} \quad (\text{B.15})$$

**Numerical example** We shall calculate the Laplace transform of a rectangular pulse (see Fig. B.9). The pulse is first decomposed into two unit step functions, one starting at  $t = 0$  and the other being delayed by the time interval  $\tau$ . The Laplace transform of the first unit function is given by  $1/s$ , that of the second by  $-(1/s) \cdot e^{-s\tau}$ . Hence, the Laplace transform of the pulse becomes

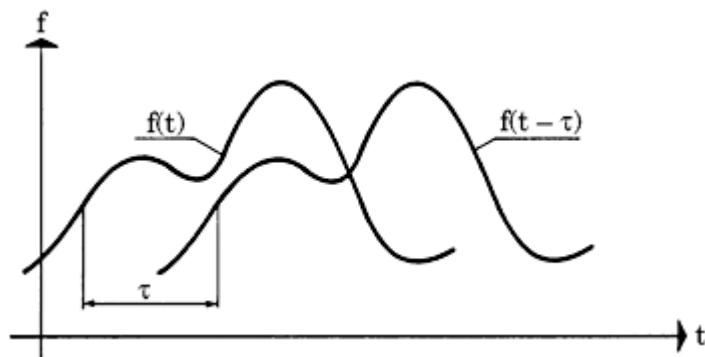
$$F(s) = \frac{1 - e^{-s\tau}}{s}$$

It is interesting to see that the Laplace transform describes a function with one single expression, whereas three separate equations would be required to describe the pulse in the time domain

$$f(t) = \begin{cases} 0 & t > t \\ 1, & 0 \leq t \leq \tau \\ 0 & t > \tau \end{cases}$$

## Differentiation and integration in the time domain

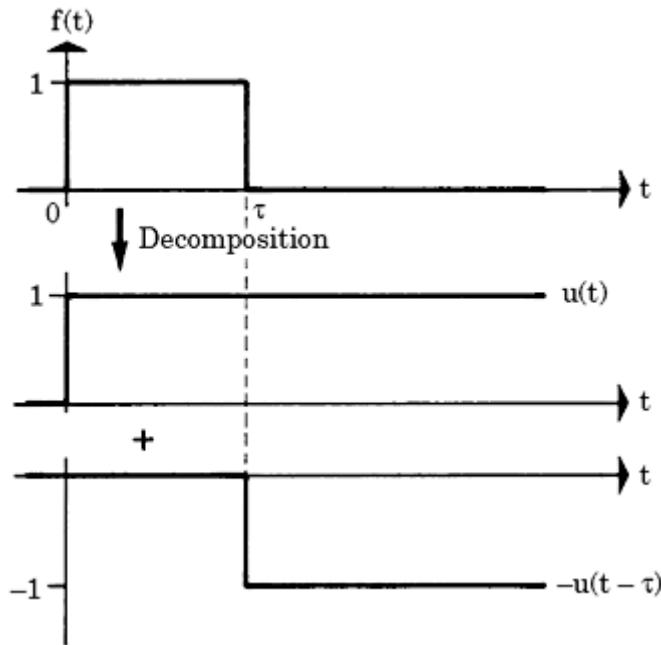
A signal  $f(t)$  and its Laplace transform  $F(s)$  are given. In many cases, we are interested to know the Laplace transform of the derivative  $df/dt$  or of the integral



**Figure B.8** Plot of a function  $f(t)$  displaced by a time delay  $\tau$ .

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.



**Figure B.9** If the Laplace transform of a single square-wave pulse (top trace) must be calculated, the pulse is first decomposed into two unit step functions (center and bottom traces).

$\int_0^t f(t) dt$ . The Laplace transform of the derivative is<sup>3,37</sup>

$$L\left[\frac{df(t)}{dt}\right] = sF(s) - f(0) \quad (\text{B.16})$$

where  $f(0)$  is the value of the signal  $f(t)$  at  $t = 0$ . A differentiation in the time domain corresponds to a multiplication by  $s$  in the complex frequency domain. The second term in Eq. (B.16) can be dropped if  $f(0)$  is zero.

A similar rule applies to integration. The Laplace transform of the integral of a function  $f(t)$  is<sup>3</sup>

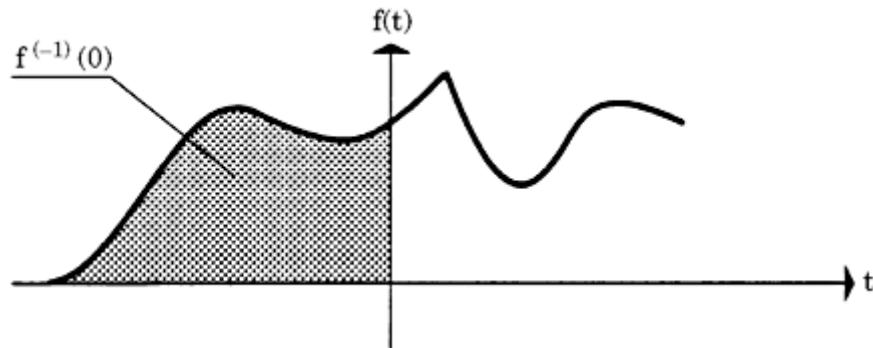
$$L\left[\int_0^t f(t) dt\right] = \frac{F(s)}{s} + \frac{f^{(-1)}(0)}{s} \quad (\text{B.17})$$

The term  $f^{(-1)}(0)$  is, by definition, the integral of  $f(t)$  over the time interval between  $-\infty$  and 0:

$$f^{(-1)}(0) = \int_{-\infty}^0 f(t)dt$$

$f^{(-1)}(0)$  is equal to the shaded area in [Fig. B.10](#). If  $f(t)$  is zero for  $t < 0$ , the second term in [Eq. \(B.17\)](#) can be dropped. Integration in the time domain corresponds to a division by  $s$  in the complex frequency domain.

Let us now apply the rule of integration to an example ([Fig. B.11](#)). This figure shows a unit step function  $u(t)$  and its first and second integrals. The first integral is a ramp function given by  $f(t) = t$  in the time domain, and the second integral is a parabola given by  $f(t) = t^2/2$ . Because  $f(t) = 0$  for  $t < 0$  in



**Figure B.10** The term  $f^{(-1)}(0)$  is given by the shaded area under the curve  $f(t)$ .

the case of the unit step function,  $f^{(-1)}(0)$  is zero. The Laplace transform of the unit step has been shown to be  $F(s) = 1/s$  (see [Sec. B.3](#) or [Table B.1](#)). The Laplace transform of the ramp function (second row in [Fig. B.11](#)) is therefore obtained by a division by  $s$ :

$$F(s) = \frac{1}{s^2}$$

Finally, the Laplace transform of the parabola (third row in [Fig. B.11](#)) is given by

$$F(s) = \frac{1}{s^3}$$

Let us now work out an example of the differentiation rule [refer to [Eq. \(B.16\)](#)]. The unit step function (see, for example, [Fig. B.7](#)) has been used throughout this text. Its Laplace transform has been shown to be  $F(s) = 1/s$ . But what is the first derivative of the step function, and what is its Laplace transform? The first derivative of the unit step is called the *delta function*  $\delta(t)$  (or impulse function).

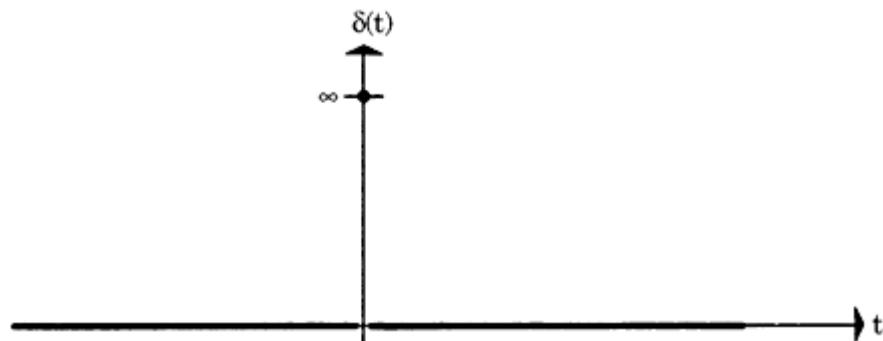
Oscillogram	Type of signal	Original function $f(t)$	Laplace transform $F(s)$
	Step function	$f(t) = u(t)$	$F(s) = \frac{1}{s}$
	Ramp	$f(t) = t \quad (t \geq 0)$	$F(s) = \frac{1}{s^2}$
	Parabola	$f(t) = \frac{t^2}{2} \quad (t \geq 0)$	$F(s) = \frac{1}{s^3}$

**Figure B.11** An example of applying the rule of integration. The Laplace transforms of a unit

step, a ramp, and a parabola are determined.

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.



**Figure B.12** A plot of the delta function  $\delta(t)$ .

For  $t < 0$  the unit step function is 0. Hence, its derivative is also 0 in this range. For  $t > 0$ , the unit step function is 1. Its derivative must therefore also be 0. At  $t = 0$ , the unit step function shows a transient from 0 to 1. Consequently, its derivative, the delta function, must be infinite at  $t = 0$ . The delta function  $\delta(t)$  is shown in [Fig. B.12](#).

The amplitude of the delta function is  $\infty$ , while the pulse width is 0. The area under the delta function is obtained by multiplying the amplitude by the pulse width. The result must be unity because the area of  $\delta(t)$  must be equal to the amplitude of the unit step function. An impulse of infinite amplitude and zero pulse width is hard to imagine, of course. But there is another way to better understand what a delta function really is ([Fig. B.13](#)).

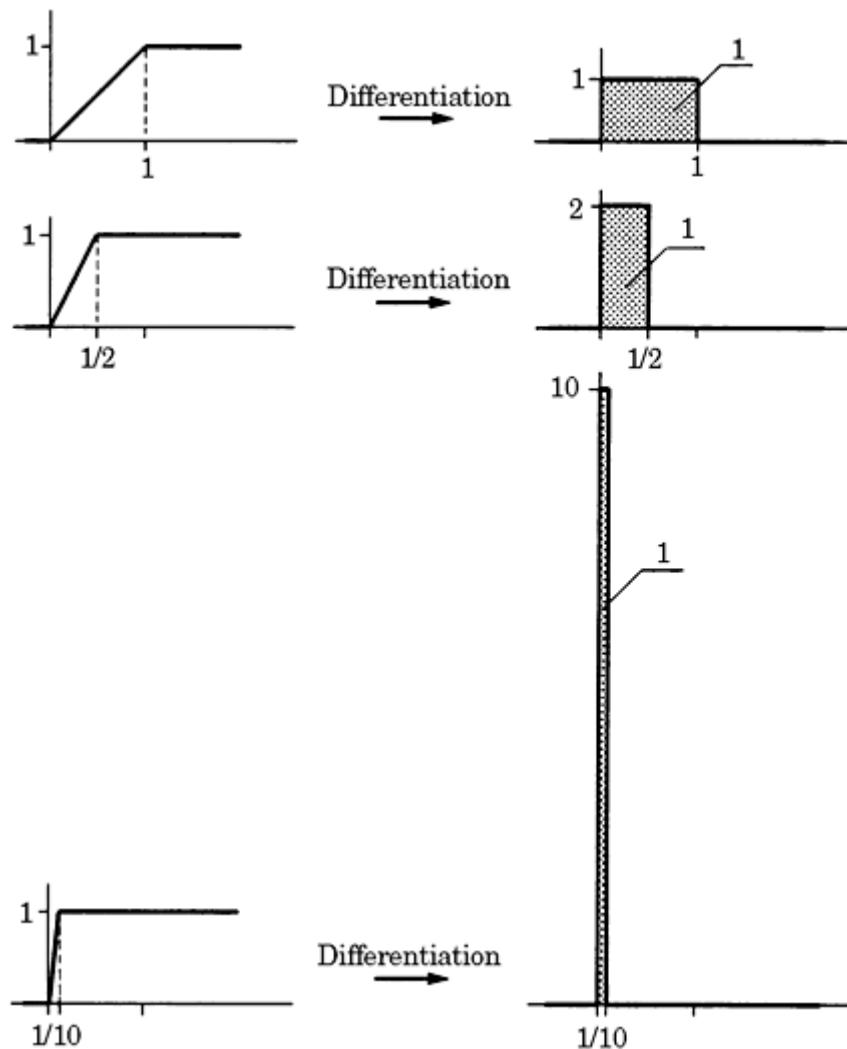
The first row of this figure shows a flattened unit step function. Its amplitude is still 1, but it takes a time of one unit (such as one second) to reach this amplitude. Consequently, the derivative of this function is an impulse having an amplitude of 1 and a pulse width of 1. The area under the pulse is 1. Assume now that our unit step becomes a little steeper (second row in [Fig. B.13](#)); therefore, it rises from 0 to 1 within 0.5 units of time. The derivative of this function is an impulse having an amplitude of 2 and a pulse width of 0.5. The area under the pulse is still 1, of course. If the unit step becomes even steeper, it may rise from 0 to 1 in as little as 0.1 units of time. Its derivative then will be a pulse having an amplitude of 10 and a pulse width of 0.1. Of course, the area of the pulse still is 1. This process can be continued as long as desired, until we finally end up with a pulse of infinite amplitude, a duration of zero, and the area under the peak is still 1.

What is the Laplace transform of the delta function now? The answer is very simple, because we only have to multiply the Laplace transform of the unit step function by  $s$ . The Laplace transform of the unit step has been shown to be

$$F(s) = \frac{1}{s}$$

Hence, the Laplace transform of the delta function is

$$F(s) = 1$$



**Figure B.13** A practical approximation of the delta function.

### The initial- and final-value theorems

In many cases where the Laplace transform  $F(s)$  of a signal  $f(t)$  is given, we are interested in knowing only the *initial value*  $f(0)$  or the *final value*  $f(\infty)$  of  $f(t)$ . The initial and final values  $f(0)$  and  $f(\infty)$ , respectively, can be obtained immediately from the Laplace transform  $F(s)$  without performing the inverse Laplace transform. The initial- and final-value theorems are given here without proof.<sup>3</sup> The initial-value theorem reads

$$f(0) = \lim_{s \rightarrow \infty} sF(s) \quad (\text{B.18})$$

And the final-value theorem reads

$$f(\infty) = \lim_{s \rightarrow 0} sF(s) \quad (\text{B.19})$$

A numerical example is given to illustrate these theorems. The Laplace transform of a (hitherto) unknown signal  $f(t)$  is given by

$$F(s) = \frac{1}{s(1 + sT)}$$

where  $T$  is a time constant. What are the values of  $f(0)$  and  $f(\infty)$ ? Using the initial-value theorem [Eq. (B.18)], we get

$$f(0) = \lim_{s \rightarrow \infty} sF(s) = \lim_{s \rightarrow \infty} \frac{s}{s(1 + sT)} = 0$$

On the other hand, the final value, according to Eq. (B.19), is

$$f(\infty) = \lim_{s \rightarrow 0} sF(s) = \lim_{s \rightarrow 0} \frac{s}{s(1 + sT)} = 1$$

## Using the Table of Laplace Transforms

Table B.1 lists the Laplace transforms of the most commonly used signals  $f(t)$ . The table also includes some of the most important theorems of the Laplace transform. When using the table, we should be aware that the Laplace transform  $F(s)$  was obtained by integrating the Laplace integral of Eq. (B.7) over the time interval  $0 \leq t \leq \infty$ . The values of the signal  $f(t)$  at negative  $t$  therefore did not contribute to  $F(s)$ . It is equivalent to state that  $f(t)$  is effectively 0 for negative  $t$ .

This fact has an effect on the inverse Laplace transform. Performing the inverse Laplace transform for a given function  $F(s)$  yields signal values  $f(t)$  for positive  $t$  only. If the Laplace transform of a signal  $f(t)$  is given by, say

$$F(s) = \frac{1}{s + a}$$

the table gives the corresponding signal  $f(t)$  as  $f(t) = e^{-at}$ . This holds true for positive  $t$  only. For negative  $t$ ,  $f(t) = 0$  by definition.

## Applying the Laplace Transform to Electric Networks

The Laplace transform is the most effective tool for analyzing the transient response of electric networks. All linear electric devices, from electric motors to operational amplifiers, are modeled by a configuration of passive elements (resistor  $R$ , inductor  $L$ , and capacitor  $C$ ) and active elements (voltage and current sources, controlled voltage, and current sources). The transient response of such electric networks is analyzed in the time domain by writing the differential equations for the branch currents and voltages. Voltages and currents in the  $R$ ,  $L$ , and  $C$  elements are related by Ohm's law as follows: For resistors:

$$u(t) = Ri(t) \quad (B.20a)$$

For inductors:

$$u(t) = L \frac{di}{dt} \quad (B.20b)$$

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

And for capacitors:

$$u(t) = \frac{1}{C} \int_0^t i dt \quad (\text{B.20c})$$

When analyzing a network in the complex frequency domain [using the rules of differentiation, [Eq. \(B.16\)](#), and of integration, [Eq. \(B.17\)](#)], we obtain, for resistors

$$U(s) = RI(s) \quad (\text{B.21a})$$

for inductors

$$U(s) = L[sI(s) - i(0)] \quad (\text{B.21b})$$

and for capacitors

$$U(s) = \frac{1}{C} \left[ \frac{I(s)}{s} + \frac{i^{(-1)}(0)}{s} \right] \quad (\text{B.21c})$$

If the initial current  $i(0)$  in an inductor  $L$  is zero, the second term in [Eq. \(B.21b\)](#) is also zero. In this case, we have  $U(s) = sL/(s)$  and the quotient

$$\frac{U(s)}{I(s)} = sL \quad (\text{B.22a})$$

can be defined as the *impedance* of the inductor. If the Laplace transform is replaced by the Fourier transform,  $s$  is replaced by  $j\omega$ , and this expression becomes

$$\frac{U(\omega)}{I(\omega)} = j\omega L$$

which is the familiar AC impedance of the inductor known from the theory of alternating currents.

If the initial charge  $i^{(-1)}/C$  in a capacitor  $C$  is zero, the second term in [Eq. \(B.21c\)](#) is also zero. In this case, we have

$$U(s) = \frac{1}{sC} I(s)$$

and the quotient

$$\frac{U(s)}{I(s)} = \frac{1}{sC} \quad (\text{B.22b})$$

is defined as the *impedance* of the capacitor. If the Laplace transform is again replaced by the Fourier transform,  $s$  is replaced by  $j\omega$ , and [Eq. \(B.22b\)](#) becomes

$$\frac{U(\omega)}{I(\omega)} = \frac{1}{j\omega C}$$

which is the familiar AC impedance of the capacitor known from the theory of alternating currents.

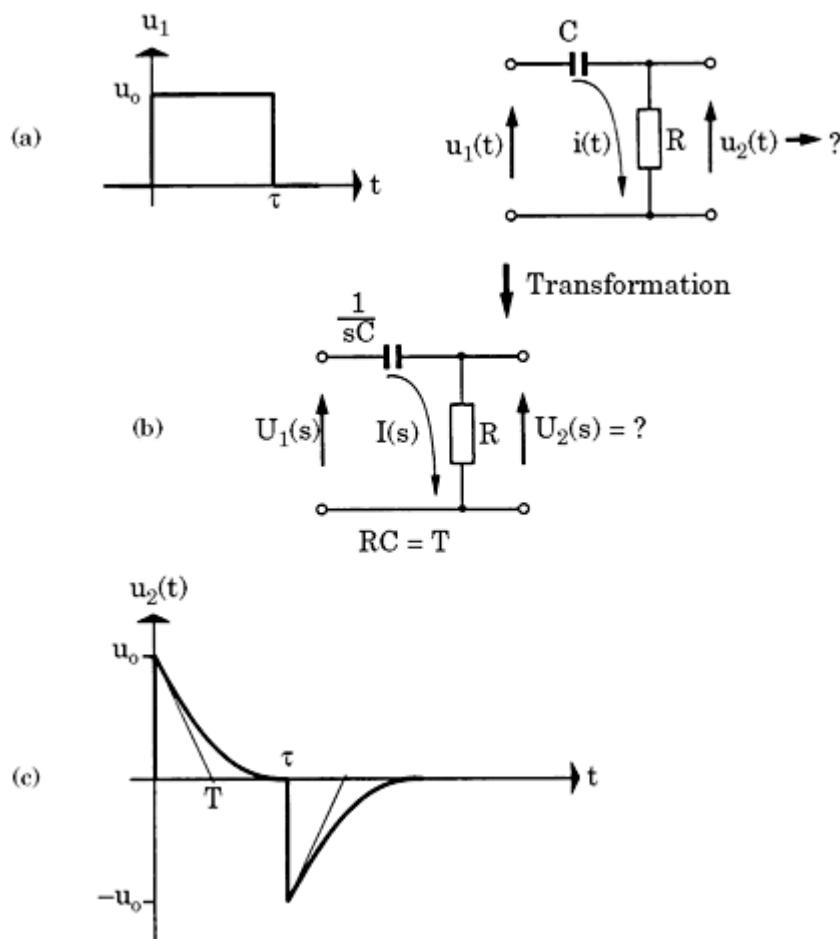
Let's now apply the Laplace transform to the analysis of a simple electric network.

**Numerical Example Transient Response of the Passive RC "Differentiator"** We want to find the transient response  $u_2(t)$  of the differentiator in Fig. B.14a on a single square-wave pulse  $u_1(t)$ . First, we introduce the variables in the time domain on the right-hand side of Fig. B.14a. These variables are transformed into the complex frequency domain in Fig. B.14b. It is assumed there is no initial charge on the capacitor.

**Solution** We can now write the node and mesh equations of the network directly in the complex frequency domain. Making use of Eqs. (B.21a) and (B.22b), we have

$$U_1(s) = RI(s) + \frac{1}{sC}I(s)$$

$$U_2(s) = RI(s)$$



**Figure B.14** Calculating the transient response of the passive  $RC$  differentiator using the Laplace transform. (a) Defining the variables in the time domain. (b) Introducing variables in the complex frequency domain. (c) Plot of the output signal  $u_2$  against

time.

---

**Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.**

After the elimination of  $I(s)$ , we obtain

$$U_2(s) = \frac{sRC}{1 + sRC} U_1(s)$$

This can be written as

$$U_2(s) = F(s)U_1(s) \quad (\text{B.23})$$

where  $F(s)$  is the transfer function of the network. If  $F(s)$  is known, the transient response of the network in any input signal  $u_1(t)$  may be calculated. In our example,  $u_1(t)$  is a single square-wave pulse whose Laplace transform was previously determined in [Sec. B.4.4](#). For  $U_1(s)$ , we can therefore write

$$U_1(s) = u_0 \frac{1 - e^{-s\tau}}{s}$$

where  $\tau$  is the duration of the pulse and  $u_0$  is its amplitude. Then,  $U_2(s)$  becomes

$$U_2(s) = u_0 \frac{1 - e^{-s\tau}}{s + 1/T}$$

where  $T = RC$ .

To transform this expression back into the time domain, we decompose it as follows:

$$U_2(s) = u_0 \frac{1}{s + 1/T} - u_0 \frac{e^{-s\tau}}{s + 1/T} \quad (\text{B.24})$$

From the table of Laplace transforms ([Table B.1](#)), the signal corresponding to the first term of [Eq. \(B.24b\)](#) is

$$u_{21}(t) = u_0 e^{-(t-\tau)/T} \quad (\text{B.25a})$$

The second term in [Eq. \(B.24b\)](#) corresponds to a decaying exponential function delayed by  $\tau$

$$u_{22}(t) = -u_0 e^{-(t-\tau)/T} \quad (\text{B.25b})$$

Because the signal  $u_{21}(t)$  is zero for  $t < 0$ , the *delayed* signal  $u_{22}(t)$  is defined only for  $t \geq \tau$ , but is zero for  $t < \tau$ . Therefore, for the combined output signal  $u_2(t)$  of the differentiator, we obtain

$$u_2(t) = \begin{cases} 0 & t < 0 \\ u_0 e^{-t/T} & 0 \leq t \leq \tau \\ u_0 [e^{-t/T} - e^{-(t-\tau)/T}] & t > \tau \end{cases}$$

The waveform of  $u_2(t)$  is plotted in [Fig. B.14c](#).

## Closing the Gap Between the Time Domain and the Complex Frequency Domain

As demonstrated in the previous section, the Laplace transforms of input and output signals of any linear electric network are related by the transfer function  $F(s)$  [cf. [Eq. \(B.23\)](#)]

$$U_2(s) = U_1(s)F(s)$$

[Equation \(B.23\)](#) enables us to determine the transient response of the network for any input signal  $u_1(t)$ .

Assume now that the network is excited by a delta function,  $u_1(t) = \delta(t)$ . The transient response of the network on a delta function  $\delta(t)$  will be hereafter denoted by  $u_2(t) = h(t)$ . As shown in [Sec. B.4.5](#), the Laplace transform of the delta function is

$$L[\delta(t)] = 1$$

Introducing this expression into [Eq. \(B.23\)](#), we obtain

$$U_2(s) = F(s)$$

From this expression, we learn that the transfer function  $F(s)$  of an electric network is the Laplace transform of the transient response  $h(t)$  on a delta function

$$F(s) = L[h(t)] \quad (\text{B.26})$$

A practical example will clarify the correspondence given by [Eq. \(B.26\)](#).

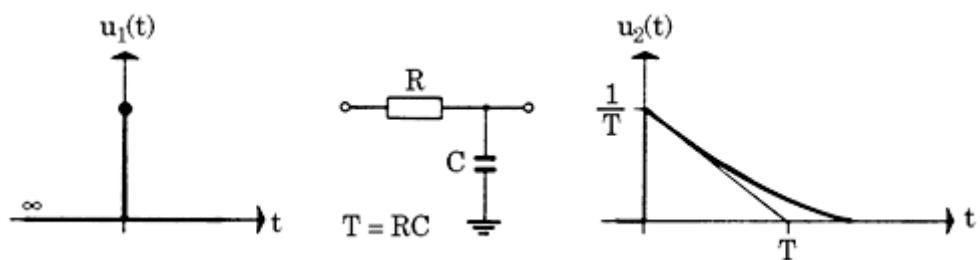
**Numerical Example** A passive  $RC$  low-pass filter (commonly called an  $RC$  integrator), as shown in [Fig. B.15](#), is excited by a delta function

$$u_1(t) = \delta(t)$$

What is the transient response  $u_2(t)$  on this input signal?

**Solution** Applying Ohm's law to the resistor and capacitor [Eqs. [\(B.21a\)](#) and [\(B.22b\)](#)], we obtain for the transfer function

$$F(s) = \frac{U_2(s)}{U_1(s)} = \frac{1}{1 + sT}$$



**Figure B.15** Calculating the transient response of the passive  $RC$  integrator using the Laplace transform.

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
 Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
 Any use is subject to the Terms of Use as given at the website.

where  $T = RC$ . According to [Eq. \(B.26\)](#), the transient response of the integrator on a delta function applied to its input is

$$u_2(t) = L^{-1}[F(s)] = L^{-1}\left[\frac{1}{1 + sT}\right]$$

Using the table of Laplace transforms ([Table B.1](#)), we obtain

$$u_2(t) = h(t) = \frac{1}{T}e^{-t/T}$$

This is the impulse response plotted on the right-hand side of [Fig. B.15](#).

Some readers will have observed that the impulse response  $h(t)$  calculated in the previous example has the dimension of  $s^{-1}$  instead of the expected  $V$ . This stems from the definition of the delta function, which is the time derivative of a unit step and has the dimension  $s^{-1}$  in fact. To get the correct physical unit, we would have to multiply the delta function by a factor  $U_0$

- $T_0$ , where  $U_0 = 1 \text{ V}$  and  $T_0 = 1 \text{ s}$ . For simplicity, this factor is omitted here and in the following calculations.

## Networks with Nonzero Stored Energy at $t = 0$

We now apply the Laplace transform to electric networks having either an inductor in which a nonzero current  $i(0)$  flows at  $t = 0$  or a capacitor on which there is a nonzero voltage  $u(0)$  at  $t = 0$ . In both cases, nonzero initial energy is stored in the network at  $t = 0$ . Let us again calculate the transient response of the  $RC$  integrator in [Fig. B.15](#) on a delta function, assuming now that the initial voltage across the capacitor is  $u(0) \neq 0$ .

Two equations in the time domain can be written for the  $RC$  integrator:

$$\begin{aligned} u_1(t) &= i(t)R + u_2(t_0) \\ u_2(t) &= \frac{1}{C} \int_0^t i dt \end{aligned}$$

Applying the Laplace transform to these equations [refer to [Eqs. \(B.21\)](#)], we obtain

$$\begin{aligned} U_1(s) &= I(s)R + U_2(s) \\ U_2(s) &= \frac{1}{C} \left[ \frac{I(s)}{s} + \frac{i^{(-1)}(0)}{s} \right] \end{aligned}$$

We can eliminate  $I(s)$  from the first of these equations. We then get

$$U_2(s) = \frac{1}{C} \left[ \frac{U_1(s) - U_2(s)}{Rs} + \frac{i^{(-1)}(0)}{s} \right]$$

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

After some manipulation, we have

$$U_2(s) = U_1(s) \frac{1}{1 + sT} + \frac{i^{(-1)}(0)}{C} \frac{T}{1 + sT}$$

where  $T = RC$  and  $i^{(-1)}(0)$  is the integral of current that flowed into the capacitor in the time interval  $-\infty < t < 0$ —meaning,  $i^{(-1)}(0)$  is the initial *charge* stored in the capacitor at  $t = 0$ . Hence,  $i^{(-1)}(0)/C$  is simply the initial voltage  $u(0)$  across the capacitor. Consequently, we obtain

$$U_2(s) = U_1(s) \frac{1}{1 + sT} + u(0) \frac{T}{1 + sT}$$

Because  $u_1(t)$  has been assumed to be a delta function,  $U_1(s) = 1$ , and we have

$$U_2(s) = \frac{1}{1 + sT} + u(0) \frac{T}{1 + sT}$$

Transforming this equation back into the time domain (see [Table B.1](#)), we obtain for  $u_2(t)$

$$u_2(t) = \frac{1}{T} e^{-ut/T} + u(0) e^{-ut/T}$$

Note that the first term is identical with the response of the  $RC$  integrator obtained for zero initial voltage across the capacitor. The second term is due to the initial charge stored in the capacitor. (Here again, to get the correct unit, we would have to multiply the first term of  $u_2(t)$  by the factor  $U_0 \cdot T_0$ , where  $U_0 = 1$  V and  $T_0 = 1$  s.)

## Analyzing Dynamic Performance by the Pole-Zero Plot

The transfer function of any linear network built from lumped elements such as resistors, inductors, capacitors, and amplifiers is given by a rational function of  $s$ :

$$F(s) = \frac{a_n s^n + a_{n-1} s^{n-1} + \dots + a_1 s + a_0}{b_m s^m + b_{m-1} s^{m-1} + \dots + b_1 s + b_0}, \quad m \geq n \quad (\text{B.27a})$$

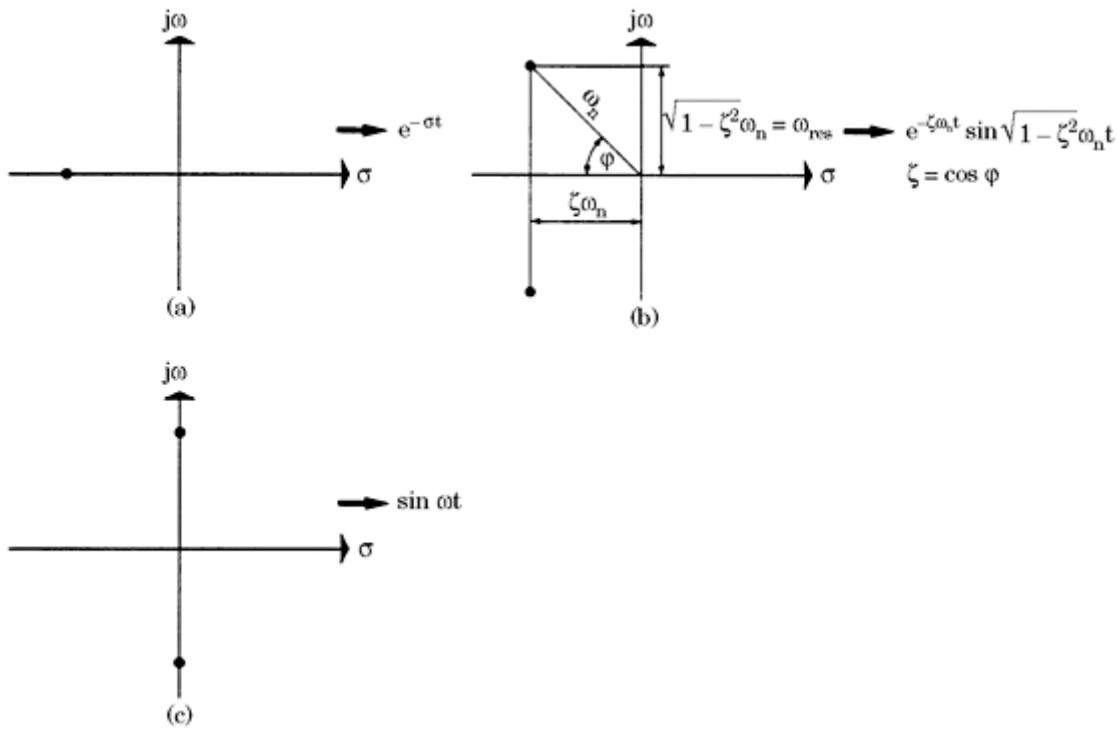
This transfer function can also be written in the factored form:

$$F(s) = \frac{(s - \alpha_1)(s - \alpha_2) \dots (s - \alpha_n)}{(s - \beta_1)(s - \beta_2) \dots (s - \beta_m)} \quad (\text{B.27b})$$

Here, the  $\alpha_i$  values are the zeroes, and the  $\beta_i$  values are the poles of the transfer function.

The  $\alpha_i$  and  $\beta_i$  can be either real or complex. For example, a real value of  $\beta_i$  corresponds to a

pole located on the real axis ( $\sigma$ -axis) in the complex  $s$ -plane (refer to [Fig. B.16](#)). If a complex pole is located at  $\beta_i = A + jB$ , another conjugate



**Figure B.16** Calculating the transient response of first- and second-order systems from the pole positions. (a) A first-order system, with the pole on the negative  $\sigma$ -axis. (b) A second-order system, with a complex-conjugate pole pair in the negative half-plane (damped oscillation). (c) A second-order system, poles on the imaginary axis (undamped oscillation).

complex pole will exist at  $\beta_i^* = A - jB$ , where the asterisk denotes the conjugate value of  $\beta_i$ . Hence, complex poles always exist as pairs of conjugate complex poles.

We will now see that the transient response of a network is very easily found if the locations of the poles and zeroes of the network transfer function  $F(s)$  are known. To transform Eq. (B.27b) back into the time domain, it is most convenient to decompose this expression into partial fractions;

$$\begin{aligned}
 F(s) &= \underbrace{\frac{R_1}{s - \beta_1} + \frac{R_2}{s - \beta_2} + \dots}_{\text{partial fractions generated by single real poles}} \\
 &\quad + \underbrace{\frac{R_i}{s - (A_i + jB_i)} + \frac{R_i^*}{s - (A - jB_i)} + \dots}_{\text{2 partial fractions generated by one pair of conjugate complex poles}}
 \end{aligned} \tag{B.27c}$$

The terms  $R_1, R_2, \dots$  are constants and are called *residues*.<sup>3,37</sup> In Eq. (B.27c), we separated two groups of partial fractions, those emanating from the single real poles, and those emanating from the pairs of conjugate complex poles.

Let us first look at the transient response  $f(t)$  due to the real poles of  $F(s)$ . The inverse Laplace transform of the term

$$F(s) = \frac{R_i}{s - \alpha_i}$$

is

$$f(t) = R_i \exp(\alpha_i t)$$

Hence, the contribution of the single real poles to the signal  $f(t)$  is

$$f(t)_{\text{single poles}} = R_1 \exp(\alpha_1 t) + R_2 \exp(\alpha_2 t) + \dots \quad (\text{B.28})$$

Note that the residues of the partial fractions due to single real poles are always real numbers. (If they are not, the signal  $f(t)$  would become complex, which is physically impossible.)

The next portion of the transient response  $f(t)$  is contributed by the complex pole pairs. For simplicity, we isolate one complex pole pair:

$$F(s)_{\text{pole pair}} = \frac{R_i}{s - (A + jB_i)} + \frac{R_i^*}{s - (A - jB_i)} \quad (\text{B.29a})$$

Here, the residues  $R_i$  and  $R_i^*$  must not necessarily be real, but can be complex. Moreover, if  $R_i$  is a complex number,  $R_i^*$  is its conjugate value—in other words, in the most general case, we have

$$\begin{aligned} R_i &= a + jb \\ R_i^* &= a - jb \end{aligned}$$

where  $a$  and  $b$  are real constants.

This is easily proved by combining the two terms in [Eq. \(B.29a\)](#) over a common denominator.

$$F(s)_{\text{pole pair}} = \frac{s(R_i + R_i^*) - A(R_i + R_i^*) + jB(R_i - R_i^*)}{s^2 - 2As + (A^2 + B^2)} s \quad (\text{B.29b})$$

All individual coefficients in the numerator of [Eq. \(B.29b\)](#) must be real—that is

$$R_i + R_i^* \rightarrow \text{real}$$

$$j(R_i - R_i^*) \rightarrow \text{real or } R_i - R_i^* \rightarrow \text{imaginary}$$

These conditions are met only if  $R_i$  and  $R_i^*$  form a conjugate complex pair of numbers, as stated earlier. [Equation \(B.29a\)](#) can therefore be rewritten as

$$F(s)_{\text{pole pair}} = \frac{a + jb}{s - (A + jb)} + \frac{a - jb}{s - (A - jb)} \quad (\text{B.29c})$$

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

Referring to [Table B.1](#), we recall that the inverse Laplace transform of  $1/(s - \alpha)$  is  $f(t) = e^{\alpha t}$ . This also applies for complex  $\alpha$ . Applying this to [Eq. \(B.29c\)](#), we obtain

$$f(t)_{\text{pole pair}} = a[e^{At}e^{jBt} + e^{At}e^{-jBt}] + jb[e^{At}e^{jBt} - e^{At}e^{-jBt}] \quad (\text{B.29d})$$

Using the well-known Euler theorem

$$\begin{aligned}\sin x &= \frac{e^{ix} - e^{-ix}}{2j} \\ \cos x &= \frac{e^{ix} + e^{-ix}}{2}\end{aligned}$$

we can write

$$f(t)_{\text{pole pair}} = 2ae^{At} \cos Bt - 2be^{At} \sin Bt \quad (\text{B.29e})$$

This can be brought into the more general form

$$f(t)_{\text{pole pair}} = Ce^{At} \cos(Bt + \Phi) \quad (\text{B.29f})$$

where

$$\begin{aligned}C &= 2\sqrt{a^2 + b^2} \\ \Phi &= -\tan^{-1} \frac{b}{a}\end{aligned}$$

The results are summarized in [Table B.2](#).

These results allow a simple interpretation of the term complex frequency, introduced earlier, which will be discussed in the next section.

## A Simple Physical Interpretation of “Complex Frequency”

Refer again to the pole-zero plot in [Fig. B.16](#). A single pole located on the negative  $\sigma$ -axis ( $\alpha < 0$ ) gives rise to the decaying exponential function (see [Table B.2](#))

$$f(t) \propto e^{\alpha t}, \quad \alpha < 0$$

**TABLE B.2 Laplace Transforms of Generalized First- and Second-Order Systems**

$F(s)$	$f(t)$
	$R_i e^{\alpha t}$

$$\frac{\frac{R_i}{s - \alpha_i}}{\frac{R_i}{s - (A + jB)} + \frac{R_i^*}{s - (A - jB)}} = Ce^{At} \cos(Bt + \Phi)$$

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

This is an exponential function having a real exponent; hence,  $\alpha$  is called a *real frequency*.

A complex pole pair located on the imaginary axis [ $A = 0$  in Eq. (B.29f)] gives rise to an undamped oscillation:

$$f(t) \propto \cos Bt$$

This is an exponential function having a purely imaginary exponent; hence,  $B$  is called an *imaginary frequency*.

A pole pair located in the left half of the  $s$ -plane, with  $A < 0$  and  $B \neq 0$ , gives rise to a damped oscillation:

$$f(t) \propto e^{At} \cos(Bt + \Phi)$$

This is an exponential function having a complex exponent; hence, we speak of an oscillation with a *complex frequency*.

It has proven useful to write partial fractions generated by complex-conjugate pole pairs in the so-called normalized form. When doing so, we introduce the substitution [refer to Eq. (B.29c)]:

$$\begin{aligned} F(s)_{\text{pole pair}} &= \frac{a + jb}{s - (A + jb)} + \frac{a - jb}{s - (A - jb)} \\ &= \frac{a + jb}{s - (-\zeta\omega_n + j\sqrt{1 - \zeta^2}\omega_n)} \\ &\quad + \frac{a - jb}{s - (-\zeta\omega_n - j\sqrt{1 - \zeta^2}\omega_n)} \end{aligned} \quad (\text{B.29g})$$

Comparing the coefficients on both sides of the arrow, we obtain the equalities

$$\begin{aligned} A &= -\zeta\omega_n \\ B &= \sqrt{1 - \zeta^2}\omega_n \end{aligned} \quad (\text{B.30})$$

where  $\zeta$  and  $\omega_n$  are the damping factor and the natural frequency, respectively.

If a transfer function  $F(s)$  has a conjugate-complex pole pair located at  $A \pm jb$  in the complex  $s$ -plane, this pole pair will generate a transient response  $f(t)$  of the form

$$f(t) = \exp(-\zeta\omega_n t) \cos(\sqrt{1 - \zeta^2}\omega_n t + \Phi) \quad (\text{B.31})$$

The time constant of the decaying exponential function in Eq. (B.31) is given by  $1/(\zeta \cdot \omega_n)$ ; the frequency of the damped oscillation is given by  $\omega_{\text{res}} = \sqrt{1 - \zeta^2}\omega_n$ . If the complex pole pair is plotted in the complex  $s$ -plane, the distance of the poles from the origin is seen to be exactly  $\omega_n$ , as shown in Fig. B.16b.

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

# Digital Filter Basics

In the age of all-digital PLLs and software PLLs, digital filters are increasing in use. This appendix is a short overview on digital filter design.

## The Transfer Function $H(z)$ of Digital Filters

Analog filters are mostly described by their frequency response  $H(j\omega)$  or by their transfer function  $H(s)$ . [Note: The frequency response of a network is often denoted as  $H(j\omega)$ , but can also be written as  $H(\omega)$ .]  $H(s)$  is defined to be

$$H(s) = \frac{O(s)}{I(s)} \quad (\text{C.1})$$

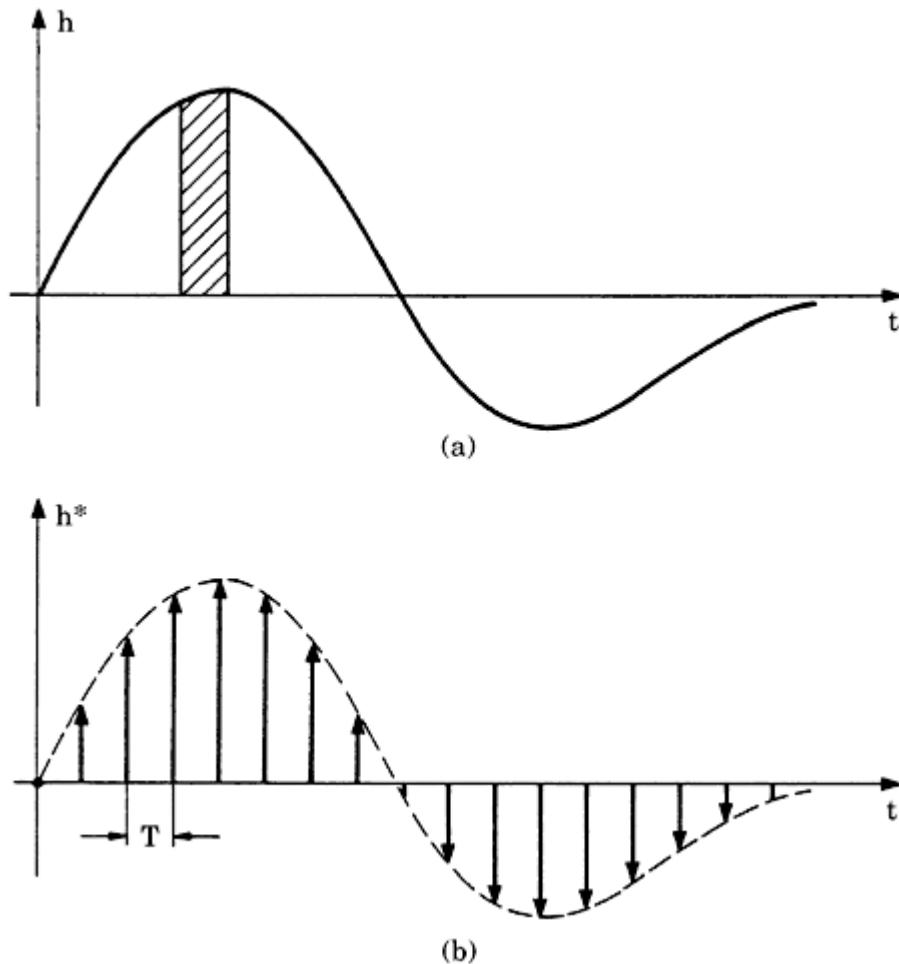
where  $O(s)$  is the Laplace transform of the output signal  $o(t)$  and  $I(s)$  is the Laplace transform of the input signal  $i(t)$ . If the input signal is a delta function

$$i(t) = \delta(t)$$

the response of the filter is called *impulse response*  $h(t)$ . Because  $I(s) = 1$  in this case (refer also to [App. B](#)), we have

$$O(s) = I(s)H(s) = H(s) \quad (\text{C.2})$$

Thus, the transfer function  $H(s)$  of the analog filter is the Laplace transform of the impulse response  $h(t)$ . This simple property is extensively used to build digital filters. Often, digital filters are designed to have an impulse response similar to that of an analog filter. This is explained by [Fig. C.1](#). [Figure C.1a](#) shows the impulse response  $h(t)$  of an analog filter; a low-pass filter has been chosen in this example. Of course,  $h(t)$  is a continuous function of time. For most analog filters, the impulse response  $h(t)$  is a damped oscillation or a decaying exponential function; in any case, the duration of the impulse response is infinite, because  $h(t)$  asymptotically approaches zero (or a constant value) for  $t \rightarrow \infty$ .



**Figure C.1** Impulse responses of analog and digital filters. (a) The impulse response  $h(t)$  of an analog filter. (b) The impulse response  $h^*(t)$  of a digital filter.  $h^*(t)$  is a sampled version of  $h(t)$ .

With a digital filter, both input and output signals are sampled signals. Its impulse response  $h^*(t)$  must therefore be a sampled signal, too. It shows up that the transfer function  $H^*(s)$  of a digital filter comes close to the transfer function  $H(s)$  of a corresponding analog filter (with some distinctions, as will be shown later) if the impulse response  $h^*(t)$  of the digital filter is a sampled version of the impulse response  $h(t)$  of the analog filter (this is illustrated in Fig. C.1b). We will see later that the sampling frequency  $f_s = 1/T$  cannot be arbitrarily chosen but must satisfy the so-called *sampling theorem*.

The sampled impulse response shown in Fig. C.1b has infinite duration as well. (It will be discussed later that the duration of the impulse response can be made finite by truncation.) For the sampled impulse response in Fig. C.1b, we can write

$$h^*(t) = T \sum_{n=0}^{\infty} h(nT) \delta(t - nT) \quad (\text{C.3})$$

where  $h(nT)$  is the amplitude of the  $n$ th sample of  $h^*(t)$ . We observe that each sample of

the impulse response has been multiplied by the sampling interval  $T$  in Eq. (C.3). This is necessary because the “area” under the delta function at sampling instant  $t = nT$  should be identical with the area under the continuous

impulse response (shaded area in Fig. C.1a) in the interval  $nT \leq t < (n + 1)T$ . If the factor  $T$  were omitted in Eq. (C.3), the dimension of the impulse response  $h^*(t)$  would be wrong. To simplify the expressions, we drop the sampling interval  $T$  in the following formulas—thus, we write  $h(n)$  for  $h(nT)$ , and so on.

Because the Laplace transform of a time-shifted delta function  $\delta(t - \tau)$  is given by  $e^{-s\tau}$ , the transfer function  $H^*(s)$  of the digital filter becomes

$$H^*(s) = T \sum_{n=0}^{\infty} h(n)e^{-snT} \quad (\text{C.4})$$

We note that the complex frequency  $s$  appears only in the form of  $e^{-snT}$ . To simplify the notation, we introduce the substitution

$$z = e^{sT} \quad (\text{C.5})$$

This is the definition of the *z-transform*. Using Eq. (C.5), the transfer function of the digital filter can be rewritten as

$$H(z) = H^*(s)|_{z=e^{sT}} = T \sum_{n=0}^{\infty} h(n)z^{-n} \quad (\text{C.6})$$

The star symbol in  $H(z)$  is dropped for simplicity, and  $H(z)$  is called the *z-transfer function* of the digital filter. We observe that the *z*-transform is nothing more than an alternative notation of the Laplace transform.

By the *z*-transform in Eq. (C.5), the complex  $s$ -plane is projected onto the complex  $z$ -plane. When working with the *z*-transform, we are performing operations in the *z-domain*. The variable  $z$  is called *z-operator*.  $z$  has a very simple physical interpretation: since  $z^{-1} = e^{-sT}$ , multiplication with  $z^{-1}$  in the *z*-domain corresponds to a time shift by one sampling interval in the time domain. Assume that  $x(t)$  is a sampled signal that only exists at the sampling instants  $t = 0, T, 2T, \dots, nT$ , and that  $X(z)$  is the corresponding *z*-transform. Then  $z^{-1}X(z)$  is the *z*-transform of the time-shifted signal  $x(t - T)$ ,  $z^{-2}X(z)$  is the *z*-transform of the time-shifted signal  $x(t - 2T)$ , and so on.

As mentioned earlier, the impulse response of the most general digital filter is an infinite series of delta functions. Digital filters having an infinite impulse response are called *IIR filters* (infinite impulse response filters). There is another class of digital filters where the impulse response is truncated when the amplitudes of the delta functions have fallen below a given threshold. This class of filters is called *FIR filters* (finite impulse response filters). Though FIR filters look simpler at first glance, the corresponding theory is more complex, so we will start the discussion with the IIR filter.

## IIR Filters

Most IIR filters are designed to perform like a known analog filter. Things are complicated by the fact that there are different approaches to transforming an analog filter into a digital. Only two of the various design procedures have

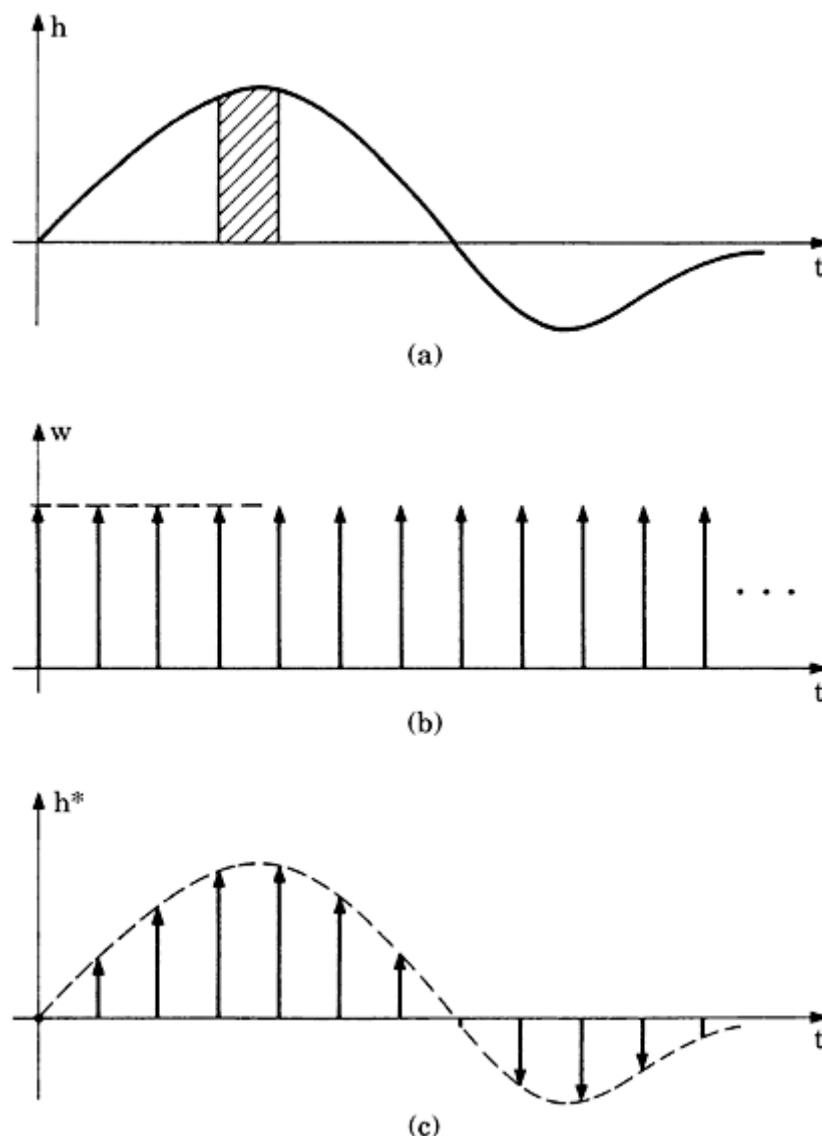
---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

survived, however: (1) the impulse-invariant  $z$ -transform, and (2) the bilinear  $z$ -transform. The second method is by far the more important, but it is easier to start with the first.

### The impulse-invariant $z$ -transform

We suppose that an IIR filter is to be built whose impulse response is a sampled version of the impulse response of a known analog filter, as shown in Fig. C.1. The transfer function of the analog filter is assumed to be  $H(s)$ . The impulse response  $h^*(t)$  of the IIR filter is therefore given by Eq. (C.3).  $h^*(t)$  is obtained by multiplying the impulse response  $h(t)$  of the analog filter by the sampling function  $w(t)$ , which is an infinite series of delta functions, all having the amplitude 1. This is illustrated by Fig. C.2. Figure C.2a shows the continuous



**Figure C.2** The impulse response of a digital filter can be obtained by multiplying the continuous impulse response  $h(t)$  of an analog filter with a sampling function  $w(t)$ . (a) The continuous impulse response  $h(t)$  of an analog filter. (b) The sampling function  $w(t)$ . (c) The result of the multiplication  $h^*(t) = h(t) \cdot w(t) \cdot T$ .

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

function  $h(t)$  again, Fig. C.2b the sampling function  $w(t)$ , and Fig. C.2c the result of the multiplication. Obviously, the transfer function  $H^*(s)$  of the IIR filter is obtained by transforming the impulse response  $h^*(t) = h(t) \cdot w(t) \cdot T$  into the  $s$ -domain. As we see from App. B (Sec. B.4.3), a multiplication of two signals corresponds to the complex convolution of their Laplace transforms in the  $s$ -domain—that is, we have

$$H^*(s) = \frac{T}{2\pi j} H(s)^* W(s) = \frac{T}{2\pi j} \oint_C H(\chi) W(s - \chi) d\chi \quad (\text{C.7})$$

In this integral,  $\chi$  is an auxiliary variable for complex frequency, and  $C$  is the contour along which the integration must be performed. First, we must know the Laplace transform  $W(s)$  of the sampling function  $w(t)$ . Applying the Laplace transform to  $w(t)$  yields

$$W(s) = 1 + e^{-sT} + e^{-2sT} + \dots \quad (\text{C.8a})$$

This is clearly a geometric series, and using the formula for the sum of the geometric series leads to

$$W(s) = \frac{1}{1 - e^{-sT}} \quad (\text{C.8b})$$

When Eq. (C.8b) is inserted into Eq. (C.7),  $H^*(s)$  becomes

$$H^*(s) = \frac{T}{2\pi j} \oint_C \frac{H(\chi)}{1 - e^{-(s-\chi)T}} d\chi \quad (\text{C.9})$$

To determine the contour  $C$  for integration, we plot the poles of  $H(\chi)$  and  $W(s - \chi)$  in the  $\chi$ -plane (Fig. C.3). The locations of the poles of  $H(\chi)$  in the  $\chi$ -plane are identical with the locations of the poles of  $H(s)$  in the  $s$ -plane. For a stable system, they must be in the left half of the  $\chi$ -plane. The poles of  $W(s - \chi)$  can be shown to be at locations  $\chi_k = s + kj\omega_s$ , where  $\omega_s = 2\pi/T$  is the angular sampling frequency and  $k$  is an integer in the range  $-\infty < k < \infty$ .<sup>37</sup>  $s$  is an arbitrary displacement here; in the example of Fig. C.3,  $s$  is assumed to be real and positive.

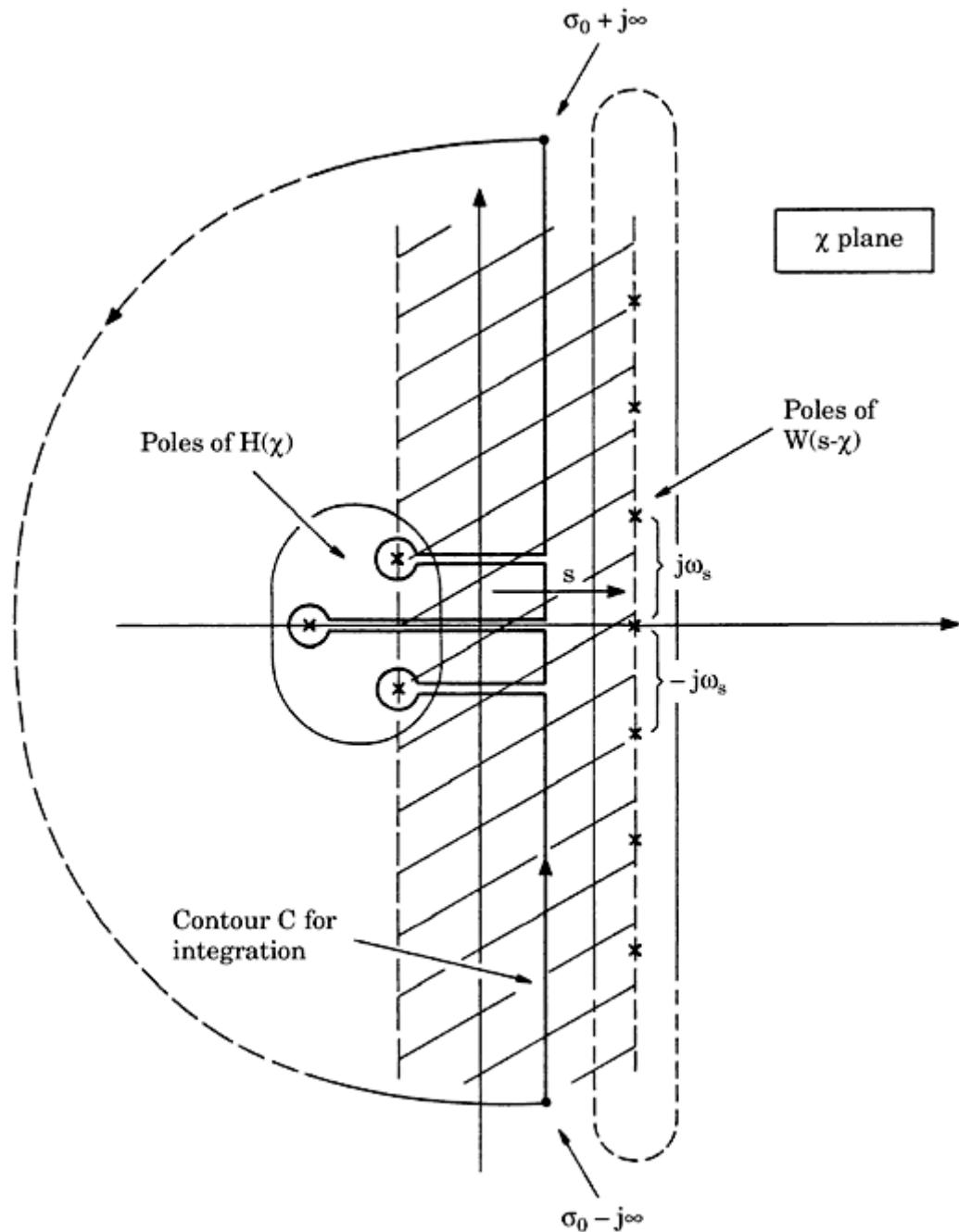
The contour  $C$  of integration must be chosen such that the integrand in Eq. (C.9) is *analytic* everywhere. According to function theory this is true when the contour is a closed curve, as shown by Fig. C.3. The vertical branch of  $C$  must be located right from the poles of  $H(\chi)$  and left from the poles of  $W(s - \chi)$ . We note that only the poles of  $H(\chi)$  are enclosed by the integration path  $C$ . The integral can be solved by applying Cauchy's residue theorem to Eq. (C.9),<sup>37</sup> and we get

$$H^*(s) = T \sum_{\substack{\text{poles} \\ \text{of } H(\chi)}} \frac{\text{Res } H(\chi)}{1 - e^{-(s-\chi)T}} \quad (\text{C.10a})$$

In Eq. (C.10a),  $\text{Res } H(\chi)$  denotes the residue of the function  $H(\chi)$  at  $\chi = \chi_0$ , where  $\chi_0$  is the location of a pole of  $H(\chi)$ . Since  $H^*(s)$  is the transfer function of

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.



**Figure C.3** Locations of the poles of  $H(\chi)$  and  $W(s - \chi)$  in the  $\chi$ -plane. The closed curve is the integration path for the convolution integral. For details, refer to the text.

a sampled system, the  $s$  operator appears only in the form of  $e^{sT}$ , so we can again substitute  $e^{sT}$  by  $z$  and obtain

(C.10b)

$$H(z) = H^*(s) \Big|_{z=e^{xT}} = T \sum_{\substack{\text{poles} \\ \text{of } H(\chi)}} \frac{\text{Res } H(\chi)}{1 - z^{-1}e^{\chi T}}$$

Here again, the star symbol in  $H(z)$  has been omitted. [Equation \(C.10b\)](#) is the definition of the impulse-variant  $z$ -transform. It enables us to calculate the  $z$ -transfer function  $H(z)$  of an IIR filter from the transfer function  $H(s)$  of a

corresponding analog filter. This computation has been done for the most important transfer functions  $H(s)$ . The result is shown in [Table C.1](#). From the definition of  $H(z)$  in [Eq. \(C.10b\)](#), it follows that the order of  $H(z)$  is identical with the order of  $H(s)$ —in other words, the impulse-invariant IIR filter has the same number of poles as the corresponding analog filter.

It follows from [Eq. \(C.10b\)](#) that the  $z$ -transfer function can always be represented as the ratio of two polynomials in  $z$ , whose order is finite—thus, we have

$$H(z) = \frac{O(z)}{I(z)} = \frac{b_0 + b_1 z^{-1} + \dots + b_M z^{-M}}{1 + a_1 z^{-1} + \dots + a_N z^{-N}} \quad (\text{C.10c})$$

The  $a_i$  and  $b_i$  ( $i = 0, 1, 2$ , and so on) are called *filter coefficients*. There is no restriction on the size of the orders  $N$  and  $M$  of the polynomials in [Eq. \(C.10c\)](#).  $M$  can be smaller, equal to, or larger than  $N$ . Taking all terms containing  $O(z)$  to the left side of the equation gives

$$O(z) + a_1 z^{-1} O(z) + \dots = b_0 I(z) + b_1 z^{-1} I(z) + \dots \quad (\text{C.10d})$$

If this is transformed back into the time domain and the sampling interval  $T$  is dropped everywhere, we get

$$o(n) = b_0 i(n) + b_1 i(n - 1) + \dots - a_1 o(n - 1) - a_2 o(n - 2) - \dots \quad (\text{C.10e})$$

which is a recursion for the output signal at sampling instant  $t = nT$ . The IIR filter is therefore also referred to as a *recursive filter*. [Equation \(C.10e\)](#) says that  $o(n)$  is calculated from a weighted sum of one or several input signals  $i(n)$ ,  $i(n - 1)$ ,  $\dots$  and from one or more terms of the output signals that have been calculated in previous sampling instants.

To see why the impulse-invariant  $z$ -transform is an inconvenient tool for the design of digital filters, we consider the frequency response of the impulse-invariant IIR filter. Let  $H(j\omega)$  be the frequency response of the analog, and  $H^*(j\omega)$  be the frequency response of the IIR filter. Then, using [Eq. \(C.7\)](#) and setting  $s = j\omega$  and  $d\chi = jd\eta$ , we have

$$H^*(j\omega) = \frac{T}{2\pi} \oint_C H(j\eta) W[j(\omega - \eta)] d\eta \quad (\text{C.11})$$

where  $\eta$  is an auxiliary variable for angular frequency. [Equation \(C.11\)](#) says that the frequency response of the IIR filter is obtained by convolving  $H(j\omega)$  with the spectrum  $W(j\omega)$  of the sampling function  $w(t)$ .

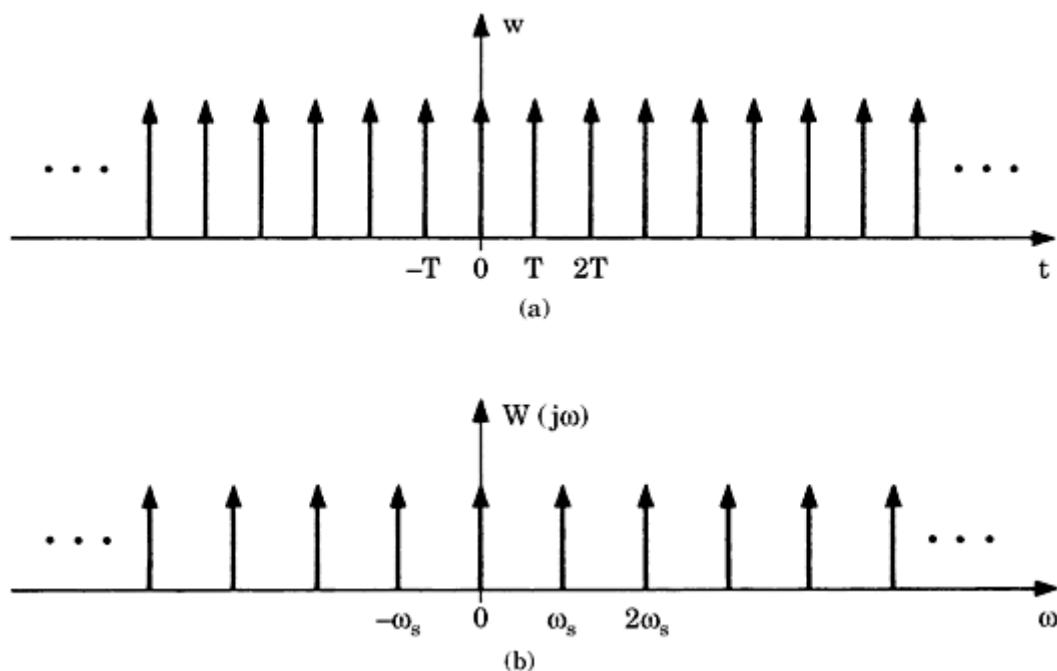
The sampling function  $w(t)$  is shown once more in [Fig. C.4a](#), and its spectrum in [Fig. C.4b](#). Let's investigate the result of the convolution by the example of [Fig. C.5](#). [Figure C.5a](#) shows the frequency response

$H(j\omega)$  of a low-pass filter. [Figure C.5b](#) presents the spectrum  $W(j\omega)$  of the sampling function  $w(t)$ . The gain of a low-pass filter is defined to be 1 at  $\omega = 0$ , and 0 at very high frequencies. The gain of the filter rolls off at frequencies higher

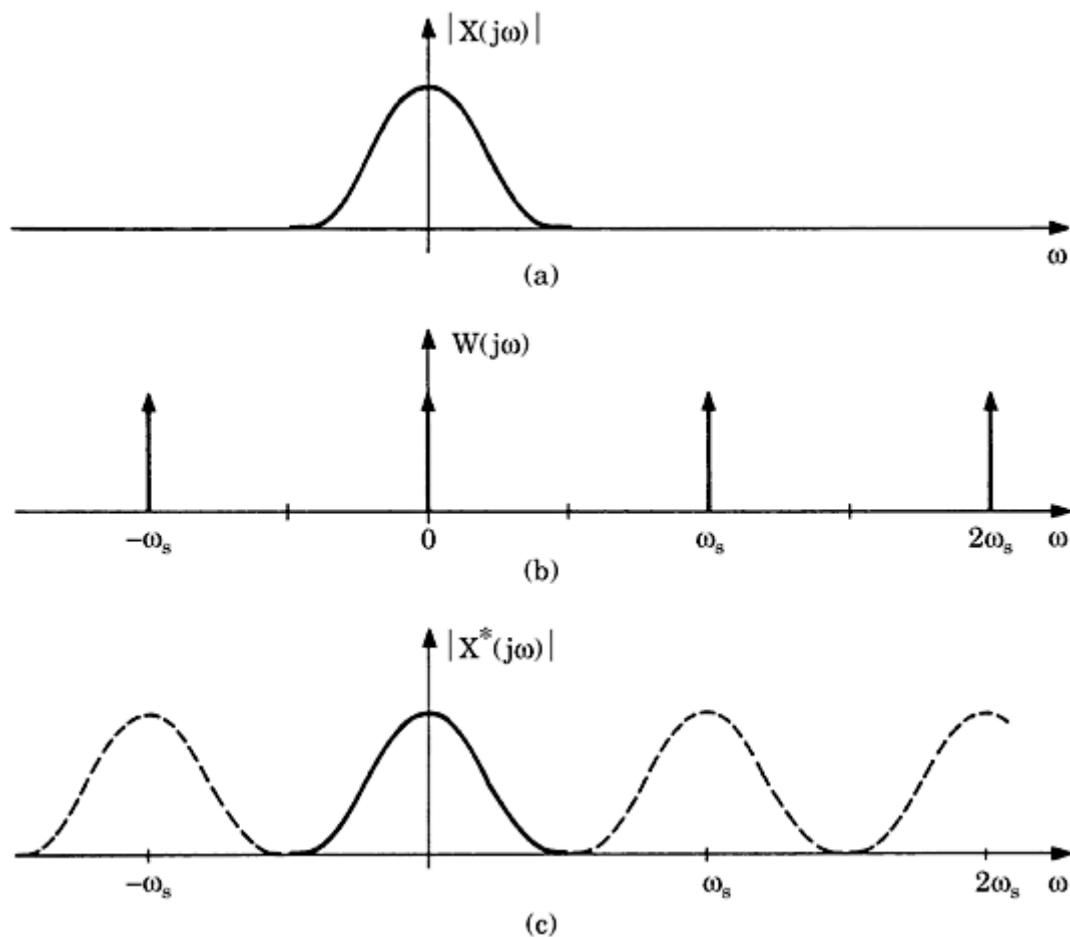
**Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.**

**TABLE C.1 Table of the Impulse-In variant z-Transform for Filters of Order 0 - 2**

Filter type	$H(s)$	
Zero-order term	1	1
Integrator	$\frac{1}{sT_i}$	$\frac{T}{T_i} \frac{1}{1 - z^{-1}}$
First-order lag	$\frac{1}{1 + s/\omega_0}$	$\omega_0 T \frac{1}{1 - z^{-1} \exp(-\omega_0 T)}$
Second-order lag	$\frac{1}{1 + 2\zeta(s/\omega_0) + (s^2/\omega_0^2)}$	$\frac{z^{-1}(\omega_0 T) \sqrt{1 - \zeta^2} \exp(-\zeta \omega_0 T) \sin(\omega_0 \sqrt{1 - \zeta^2} T)}{1 - z^{-1} 2 \exp(-\zeta \omega_0 T) \cos(\omega_0 \sqrt{1 - \zeta^2} T) + z^{-2}}$
Second-order lag, linear term in numerator	$\frac{s/\omega_1}{1 + 2\zeta(s/\omega_0) + (s^2/\omega_0^2)}$	$\frac{\omega_0^2 T}{\omega_1} \frac{1 - z^{-1} \exp(-\zeta \omega_0 T) [\cos(\omega_0 \sqrt{1 - \zeta^2} T) - \zeta \sin(\omega_0 \sqrt{1 - \zeta^2} T)]}{1 - z^{-1} 2 \exp(-\zeta \omega_0 T) \cos(\omega_0 \sqrt{1 - \zeta^2} T) + z^{-2}}$
Second-order lag, constant plus linear term in numerator	$\frac{1 + s/\omega_1}{1 + 2\zeta(s/\omega_0) + (s^2/\omega_0^2)}$	$\frac{\omega_0^2 T}{\omega_1} \frac{1 - z^{-1} \exp(-\zeta \omega_0 T) \{[\cos(\omega_0 \sqrt{1 - \zeta^2} T) - \zeta \sin(\omega_0 \sqrt{1 - \zeta^2} T)] + z^{-1} [\zeta \cos(\omega_0 \sqrt{1 - \zeta^2} T) + \sin(\omega_0 \sqrt{1 - \zeta^2} T)]\}}{1 - z^{-1} 2 \exp(-\zeta \omega_0 T) \cos(\omega_0 \sqrt{1 - \zeta^2} T) + z^{-2}}$

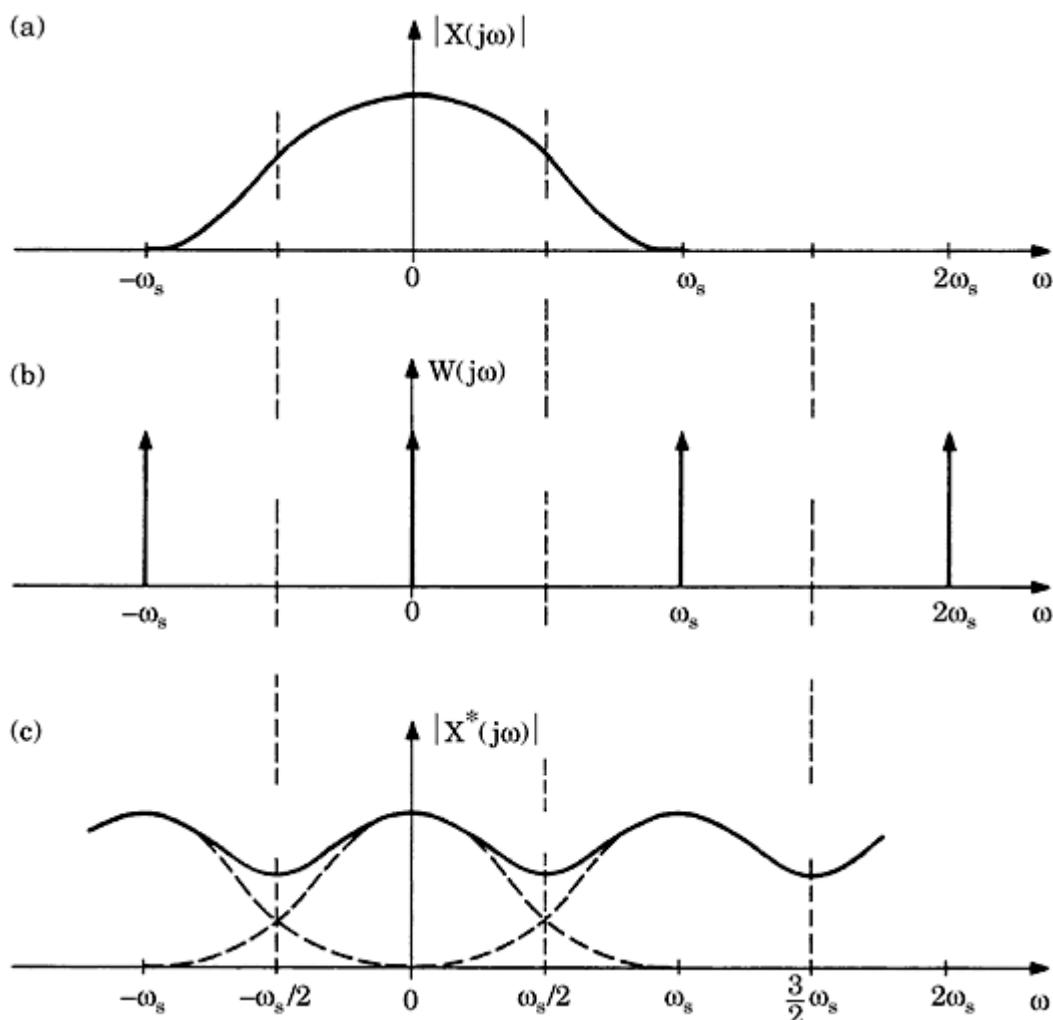


**Figure C.4** The sampling function  $w(t)$  and its spectrum  $W(j\omega)$ . (a) Sampling function  $w(t)$ .  
(b) Spectrum  $W(j\omega)$ .



**Figure C.5** The result of the convolution of  $X(j\omega)$  and  $W(j\omega)$  for the case of negligible aliasing. (a) The frequency response  $X(j\omega)$  of an analog filter. (b) Spectrum  $W(j\omega)$  of the sampling function  $w(t)$ . (c) The result of the convolution  $X(j\omega)^*W(j\omega)$ .

than the 3-dB corner frequency. In the given example, the corner frequency has been chosen such that the gain of the filter is near 0 for frequencies higher than  $\omega_n = \omega_s/2$ ;  $\omega_n$  is called the *Nyquist frequency*. The result of the convolution is shown in Fig. C.5c. Because the impulse response of the IIR filter is a sampled function, its frequency response becomes *periodic*. The period on the frequency axis is equal to the angular sampling frequency  $\omega_s$ . The solid curve in Fig. C.5c is the so-called *main lobe*, while the dashed curve represents the *side lobes* of the frequency response. Because the gain of the analog filter is nearly zero at frequencies above the Nyquist frequency, the side lobes do not markedly overlap with the main lobe of the frequency response  $H^*(j\omega)$ . We shall now consider an example, where the gain  $H(j\omega)$  has a value much greater than 0 at the Nyquist frequency (Fig. C.6a). Figure C.6b once more shows the spectrum  $W(j\omega)$  of the sampling function  $w(t)$ , and Fig. C.6c is the result of the convolution  $H(j\omega)^*W(j\omega)$ . Now the main and side lobes strongly overlap, and the transfer function of the digital filter (in the frequency range  $-\omega_s/2 \leq \omega < \omega_s/2$ ) markedly deviates from the transfer function of the analog filter. This effect is called *aliasing*. The sampling theorem (also called the Shannon or Nyquist theorem)



**Figure C.6** This is similar to Fig. C.5, but with severe aliasing. The gain  $X(j\omega)$  of the analog filter is too high at the Nyquist frequency  $\omega_n = \omega_s/2$ . (a) The frequency response  $X$

$(j\omega)$  of an analog filter. (b) Spectrum  $W(j\omega)$  of the sampling function  $w(t)$ . (c) The result of the convolution  $X(j\omega)^*W(j\omega)$ .

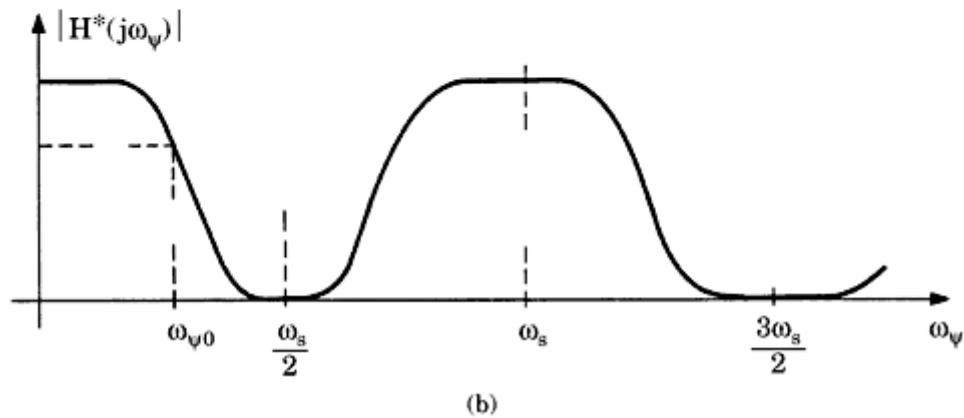
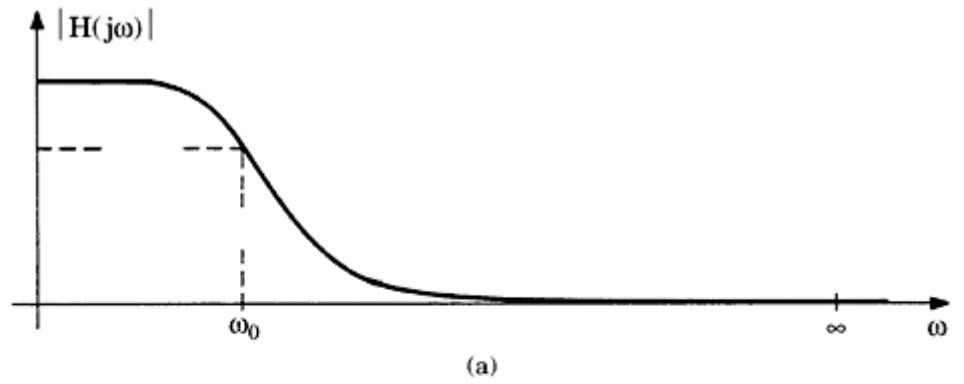
dictates that the sampling frequency must be chosen at least twice a critical frequency  $\omega_c$ , where the gain  $H(j\omega_c)$  is so low it can be neglected. A numerical example will demonstrate the implications of the sampling theorem.

**Numerical Example** Assume that we want to design a digital two-pole lowpass filter having a 3-dB corner frequency of 1 kHz. A corresponding analog lowpass filter would roll off its gain above 1 kHz with 40 dB/decade, so the filter would have an attenuation of 40 dB at 10 kHz, 80 dB at 100 kHz, and so on. If we claim that the IIR filter actually reaches an attenuation of 80 dB, its Nyquist frequency must be at least 100 kHz. Thus, the sampling frequency must be *at least* 200 kHz—that is, 200 times as high as the 3-dB corner frequency.

The requirement for very high sampling frequencies is the main drawback of the impulse-invariant IIR filter. Fortunately, the bilinear  $z$ -transform enables us to realize IIR filters, where the main and side lobes of the transfer function do not overlap and can therefore be realized with a much lower sampling frequency.

### The bilinear $z$ -transform

The bilinear  $z$ -transform also converts the transfer function  $H(s)$  of a given analog filter into the transfer function  $H(z)$  of a digital filter but uses a different mathematical procedure. [Figure C.7a](#) shows the frequency response of an



**Figure C.7** A principle of the bilinear  $z$ -transform. (a) The frequency response  $H(j\omega)$  of an analog filter. (b) The frequency axis of (a) is projected onto the pseudo-frequency axis  $\omega_\psi$ . Transforming the frequency response of the filter in (a) into the pseudo-frequency domain results in the periodic frequency response  $H^*(j\omega_\psi)$ .

analog filter; again, a lowpass filter has been chosen. Its 3-dB (angular) corner frequency is denoted  $\omega_0$ . We now introduce a new radian frequency variable, the so-called pseudo-frequency  $\omega_\psi$ . We define that the pseudo-frequency  $\omega_\psi$  and the (radian) frequency  $\omega$  are related by a tangent transform

$$\omega = \frac{2}{T} \tan \frac{\omega_\psi T}{2} \quad (\text{C.12})$$

Thus, the frequency range  $-\infty < \omega < \infty$  is projected onto the pseudo-frequency interval— $\omega_s/2 \leq \omega_\psi \leq \omega_s/2$ , which is called the *Nyquist interval*. Because the tangent transform is not unambiguous, the frequency range  $-\infty < \omega < \infty$  is also projected onto the other Nyquist intervals—that is, onto the pseudo-frequency intervals  $\omega_s/2 \leq \omega_\psi \leq 3\omega_s/2$ ,  $3\omega_s/2 \leq \omega_\psi \leq 5\omega_s/2$ , and so on. If we transform the frequency response of the analog filter into the pseudo-frequency domain, we get the periodic transfer function  $H^*(j\omega_\psi)$ , as plotted in [Fig. C.7b](#). Between  $H(j\omega)$  and  $H^*(j\omega_\psi)$ , there is the relation

$$H^*(j\omega_\psi) = H(j\omega) \Big|_{\omega=(2/T)\tan(\omega_\psi T/2)} \quad (\text{C.13})$$

which says that the gain of the new filter at pseudo-frequency  $\omega_\psi$  is identical to the gain of the analog filter at frequency  $\omega$ , where  $\omega$  and  $\omega_\psi$  are related by [Eq. \(C.12\)](#). Because the frequency response of the new filter is periodic, it is a *digital filter*. The 3-dB corner frequency of the digital filter is denoted as  $\omega_{\psi 0}$  in [Fig. C.7b](#), which is called the *pseudo corner frequency*. By the tangent transform, the pseudo corner frequency becomes smaller than the original corner frequency of the analog filter. Normally, it is required that the digital filter have the same corner frequency as the original analog filter—meaning, the pseudo corner frequency should be  $\omega_0$ . This is easily accomplished by *prewarping* the frequency response of the analog filter. If the prewarped corner frequency of the analog filter is denoted as  $\omega_{0p}$ , we have

$$\omega_{0p} = \frac{2}{T} \tan \frac{\omega_0 T}{2} \quad (\text{C.14})$$

It is easily seen then that the pseudo corner frequency becomes  $\omega_0$ .

To realize the digital filter, we must know its  $z$ -transfer function  $H(z)$ . To get  $H(z)$ , we first introduce complex frequency variables. We assume that the transfer function of the analog filter is  $H(s)$ ; its frequency response  $H(j\omega)$  is obtained simply by setting  $s = j\omega$ . In the same way, the transfer function of the digital filter is given by  $H^*(s_\psi)$ ; its frequency response  $H^*(j\omega_\psi)$  is also obtained by setting  $s_\psi = j\omega_\psi$ . To transform the complex frequency domain into the complex pseudo-frequency domain, we must now use the transformation

$$s = \frac{2}{T} \tanh \frac{s_\psi T}{2} \quad (\text{C.15})$$

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

When we set  $s = j\omega$  and  $s_\psi = j\omega_\psi$  in Eq. (C.15), Eq. (C.12) is obtained again. Using Eq. (C.15), the transfer function  $H^*(s_\psi)$  of the digital filter is calculated from

$$H^*(s_\psi) = H(s) \Big|_{s=(2/T)\tanh(s_\psi T/2)} \quad (\text{C.16})$$

Using Euler's relations, the tanh function can be written in the form of exponential functions:

$$\tanh \frac{s_\psi T}{2} = \frac{1 - e^{-s_\psi T}}{1 + e^{-s_\psi T}} \quad (\text{C.17})$$

Equation (C.16) can therefore be rewritten as

$$H^*(s_\psi) = H(s) \Big|_{s=(2/T)(1-e^{-s_\psi T})/(1+e^{-s_\psi T})} \quad (\text{C.18a})$$

Because the complex pseudo-frequency appears exclusively in the form  $\exp(ns_\psi T)$ ,  $\exp(s_\psi T)$  can be replaced by  $z$ :

$$z = e^{s_\psi T} \quad (\text{C.18b})$$

Thus, we apply the  $z$ -transform to the pseudo-frequency domain. Using the substitution of Eq. (C.18b), we finally get

$$H(z) = H(s) \Big|_{s=(2/T)(1-z^{-1})/(1+z^{-1})} \quad (\text{C.19})$$

The star symbol in  $H(z)$  has been omitted for simplicity. Equation (C.19) defines the *bilinear z-transform*. It enables us to calculate the  $z$ -transfer function  $H(z)$  of a digital filter directly from a given transfer function  $H(s)$  of an analog filter just by substituting  $s$  by

$$s = \frac{2}{T} \frac{1 - z^{-1}}{1 + z^{-1}}$$

Table C.2 shows the bilinear  $z$ -transforms for filters of order 0 to 2. As with the impulse-invariant  $z$ -transform, the order of the bilinear  $z$ -transform  $H(z)$  is identical with the order of the corresponding transfer function  $H(s)$  of the analog filter. When comparing Table C.2 with Table C.1 (impulse-invariant  $z$ -transform), we observe that the expressions are quite similar but that the numerator polynomials are different.

When the  $z$ -transfer function  $H(z)$  is transformed back into the time domain, a recursion for the output signal  $o(n)$  is obtained that has the same form as the impulse-invariant IIR filter [Eq. (C.10e)].

To see the real benefit of the bilinear  $z$ -transform, let's have a look again at Fig. C.7. The filter in this example is a lowpass filter. The gain  $H(j\omega)$  of the analog filter (Fig. C.7a) approaches 0 only for  $\omega \rightarrow \infty$ . Because of the tangent transformation, however, the gain of

the digital filter becomes *exactly* 0 at the

**TABLE C.2 Table of the Bilinear  $z$ -Transform for Filters of Order 0 – 2**

Filter type	$H(s)$
Zero-order term	1
Integrator	$\frac{1}{sT_i}$
	$\frac{T}{2T_i} \frac{1 + z^{-1}}{1 - z^{-1}}$
First-order lag	$\frac{1}{1 + s/\omega_0}$
	$\frac{1 + z^{-1}}{1 + 2/(\omega_0 T) + z^{-1}[1 - 2/(\omega_0 T)]}$
Second-order lag	$\frac{1}{1 + 2\xi(s/\omega_0) + (s^2/\omega_0^2)}$
	$\frac{1 + 2}{1 + 4\xi/(\omega_0 T) + 4(\omega_0 T)^2 + z^{-1}[2 - 8\xi/(\omega_0 T)]}$
Second-order lag, linear term in numerator	$\frac{s/\omega_1}{1 + 2\xi(s/\omega_0) + (s^2/\omega_0^2)}$
	$\frac{2}{\omega_1 T} \frac{1 + z^{-1}}{1 + 4\xi/(\omega_0 T) + 4(\omega_0 T)^2 + z^{-1}[2 - 8\xi/(\omega_0 T)]}$
Second-order lag, constant plus linear term in numerator	$\frac{1 + s/\omega_1}{1 + 2\xi(s/\omega_0) + (s^2/\omega_0^2)}$
	$\frac{(1 + 2/\omega_1 T) + 2z^{-1}}{1 + 4\xi/(\omega_0 T) + 4(\omega_0 T)^2 + z^{-1}[2 - 8\xi/(\omega_0 T)]}$

Nyquist pseudo-frequency  $\omega_c = \omega_s/2$ , as shown in Fig. C.7b. Consequently, the main and side lobes of the pseudo-frequency response  $H^*(j\omega_p)$  do not overlap, which means that this digital filter does *not exhibit aliasing*. This enables us to choose a sampling frequency much lower than that with the impulse-invariant IIR filter. The lack of aliasing is especially important for the realization of high-pass filters and differentiators. For both of these types, the gain at  $\omega = 0$  should be zero. When realized by the impulse-invariant  $z$ -transform, however, the gain of the digital filter becomes nonzero at  $\omega = 0$ , owing to aliasing. This is the reason why practically all IIR filters are designed by the bilinear  $z$ -transform.

**Numerical Example** We shall design a single-pole Butterworth lowpass digital filter. The 3-dB corner frequency of the IIR filter is required to be  $f_0 = 200$  Hz. The transfer function of the corresponding analog filter is given by

$$H(s) = \frac{1}{1 + s/\omega_0} \quad (\text{C.20a})$$

The sampling frequency is chosen as  $f_s = 1000$  Hz, which is not considerably higher than the corner frequency. First, the 3-dB corner frequency of the corresponding analog filter must be prewarped; thus, we have

$$\omega_0 = 2\pi \cdot 200 = 1256 \text{ s}^{-1}$$

$$\omega_{0p} = \frac{2}{T} \tan \frac{\omega_0 T}{2} = 1453 \text{ s}^{-1}$$

The bilinear  $z$ -transform of the digital filter is found in Table C.2 and reads

$$H(z) = \frac{1 + z^{-1}}{1 + 2/(\omega_{0p}T) + z^{-1}[1 - 2/(\omega_{0p}T)]} \quad (\text{C.20b})$$

Entering numerical values yields and normalizing the constant term in the denominator to 1, we get

$$H(z) = \frac{0.4208 + 0.4208z^{-1}}{1 - 0.1584z^{-1}}$$

Transforming back into the time domain, we get the recursion for the IIR filter

$$o(n) = 0.1584o(n - 1) + 0.4208i(n) + 0.4208i(n - 1)$$

In the most general case, the filter to be realized can have a large number of poles and zeroes. The analog filter from which the design starts then will have a transfer function of the form

$$H(s) = \frac{(1 + s/\omega_{01}) \cdot (1 + 2s\zeta_3/\omega_{03} + [s/\omega_{03}]^2) \cdot (\underline{\hspace{2cm}})}{(1 + s/\omega_{02}) \cdot (1 + 2s\zeta_2/\omega_{04} + [s/\omega_{04}]^2) \cdot (\underline{\hspace{2cm}})}$$

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

Elliptic filters for example, have transfer functions of this kind.<sup>13</sup> Their transfer function may include a real-axis zero (first term in the numerator); if this zero does not exist, the corresponding term does not show up. The second term in the numerator represents a complex-conjugate zero pair. There may be further complex-conjugate zero pairs, as indicated by the empty bracket in the numerator. In a similar way, the transfer function may have a real-axis pole (first term in the denominator) and one or several complex-conjugate pole pairs. The corresponding corner frequencies of this filter would be  $\omega_{01}, \omega_{03}, \dots, \omega_{02}, \omega_{04}, \dots$ , and so on. When an IIR filter having this transfer function must be designed, all corner frequencies in the numerator and denominator must be pre-warped using Eq. (C.14).

A user who wants to design an IIR filter today probably will no longer use  $z$ -transform tables, but rather one of the numerous software tools available now.<sup>25,35</sup>

## FIR Filters

Like IIR filters, FIR filters are often designed on the base of an existing analog filter. As we will see later, this must not necessarily be the case. Let's start again by plotting the impulse response  $h(t)$  of an existing analog filter, as shown in Fig. C.1a. The digital filter is assumed to have an impulse response, as shown in Fig. C.1b, but we now set all samples  $h(n) = 0$  for  $n \geq N$ .  $N$  is a positive integer and is also referred to as *length of the FIR filter*. The impulse response of the digital filter now reads

$$h^*(t) = T \sum_{n=0}^{N-1} h(n) \delta(t - nT) \quad (\text{C.21})$$

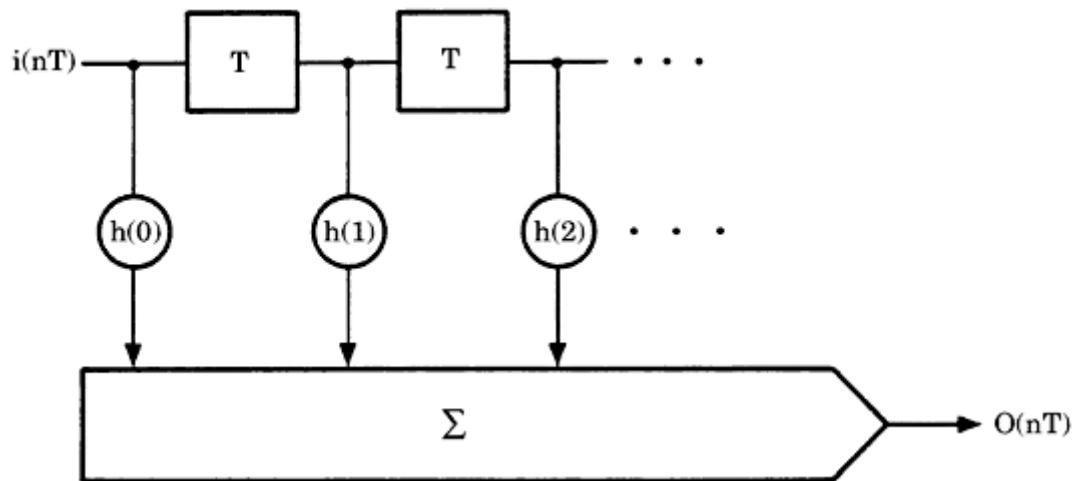
The  $z$ -transfer function of the FIR filter then becomes [Eq. (C.6)]

$$H(z) = T \sum_{n=0}^{N-1} h(n) z^{-n} \quad (\text{C.22})$$

Transforming back into the time domain, the output signal of the FIR filter at sampling instant  $t = nT$  is obtained from

$$o(n) = T \sum_{k=0}^{N-1} i(n) h(n - k) \quad (\text{C.23})$$

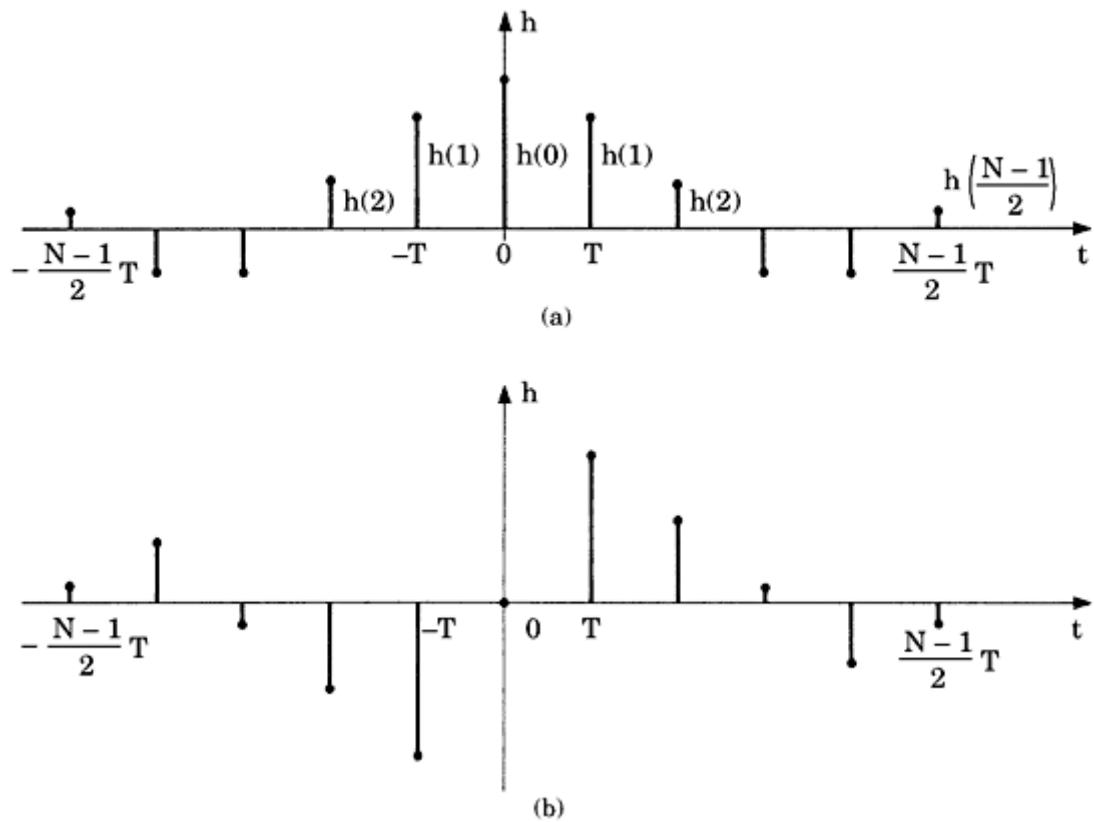
This is not a recursion, since  $o(n)$  does not depend on previously calculated samples  $o(n-1), o(n-2)$ , and so on. For this reason, FIR filters are often referred to as *nonrecursive filters*. The signal flow diagram of an FIR filter is shown in Fig. C.8; in this drawing,  $T$  has been set 1 for simplicity. Because the FIR filter looks like a tapped delay line, the filter length  $N$  is also called the *number of taps*. The filter coefficients  $h(0), h(1), \dots, h(N-1)$  are nothing more than the values of the impulse response  $h^*(t)$  at sampling times  $t = 0, T, 2T, \dots$



**Figure C.8** A signal flow diagram of the FIR filter. The blocks marked with  $T$  are delay blocks.

For the FIR filter, the coefficients  $h(n)$ ,  $n = 0 \dots N - 1$ , can be chosen arbitrarily, so the impulse response may have any desired shape. This degree of freedom was not attainable with the IIR filter. We conclude, therefore, that FIR filters can be built which do not have an analog counterpart.

Let's first consider two kinds of impulse response that are of particular interest. [Figure C.9a](#) shows a finite impulse response, which is an even function of time  $t$ ; in [Fig. C.9b](#), another impulse response is plotted, which is an odd function of time.



**Figure C.9** Two special cases of the impulse response of an FIR filter. (a) The impulse response is an even function of time. (b) The impulse response is an odd function of time.

From the theory of Fourier transform, it follows that the spectrum of a signal, which is an even function of time, is purely real.<sup>12,13,14</sup> The spectrum of a signal, however, which is an odd function of time, is purely imaginary. Because the frequency response  $H(j\omega)$  of the FIR filter is by definition the Fourier transform of its impulse response, the frequency response of an FIR filter having an even impulse response would be purely real.

This means that the FIR filter provides frequency-dependent gain with zero phase shift. Such a filter is called a *zero-phase filter*; unfortunately, it is not realizable in real-time operation. Moreover, the frequency response of an FIR filter, which has an odd impulse response, would be purely imaginary. Its phase shift would be  $90^\circ$  independent of the frequency, and the amplitude response could be any desired function of frequency. This kind of FIR filter can also be considered as a zero-phase filter; sometimes they are referred to as *zero-phase filters of the second kind*. If zero-phase filters were realizable, they could be used to build *ideal filters*. To give an example, an ideal lowpass filter would have a real frequency response  $H(j\omega)$ , which is exactly 1 within the passband, and 0 at all other frequencies. The impulse response  $h(t)$  of the ideal lowpass filter would be a sinc function

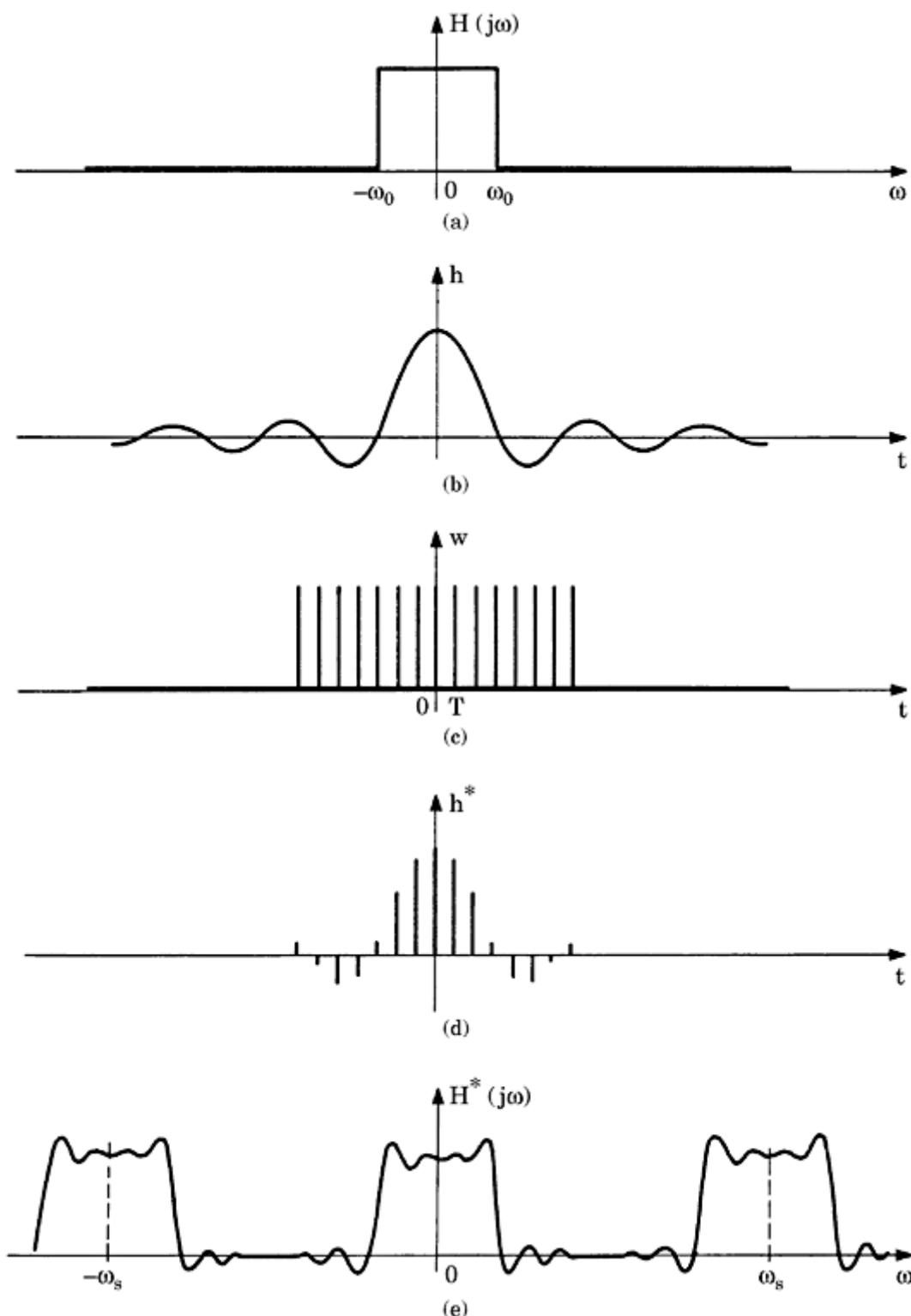
$$h(t) = \frac{\sin \omega_0 t}{\pi t} \quad (\text{C.24})$$

where  $\omega_0$  is the angular corner frequency of the ideal lowpass filter. [Note:  $\text{sinc}(x)$  is an abbreviation for  $\sin(\pi x)/(\pi x)$ .] The frequency response of the ideal lowpass filter is shown in Fig. C.10a, and the impulse response appears in Fig. C.10b. Such a filter is not realizable, of course, because the impulse response starts at  $t = -\infty$ . The filter becomes realizable to an approximation after two important modifications. First, a *window function*  $w(t)$ —as shown in Fig. C.10c—is used to cut out a finite portion of the impulse response; thus, the finite impulse response  $h^*$ —as shown in Fig. C.10d—is obtained by multiplying the continuous impulse response  $h(t)$  with the window function:

$$h^*(t) = h(t)w(t)T$$

(The multiplication with sampling interval  $T$  must be made to get the correct dimension for  $h^*(t)$  [cf. also Eq. (C.3)].) The impulse response shown in Fig. C.10d is finite. If the filter were realizable, its frequency response could be made to approximate very closely with that of the ideal lowpass filter by choosing a large number of samples  $N$ . Let  $H_{\text{zph}}(j\omega)$  be the frequency response of this (unrealizable) filter. To get a realizable filter, a second modification must be made. The impulse response of Fig. C.10d is delayed such that it starts at  $t = 0$ . If the required time delay is  $\tau$ , the frequency response of the modified FIR filter is given by

$$H_{\text{lph}}(j\omega) = H_{\text{zph}}(j\omega)e^{-j\omega\tau} \quad (\text{C.25})$$



**Figure C.10** This figure illustrates the design procedure for window FIR filters. (a) The design starts by defining the frequency response  $H(j\omega)$  of an “ideal” analog filter. (b) The impulse response  $h(t)$  of the “ideal” analog filter is calculated. (c) To get a finite and sampled impulse response, a window function  $w(t)$  is defined,

which will be used to cut out a finite portion of the impulse response  $h(t)$ . (d) Result of the multiplication of  $h(t)$  and  $w(t)$ . This is the impulse response of the FIR filter. (e)  $H^*(j\omega)$  is the frequency response of the windowed FIR filter.

The phase of term  $H_{zph}(j\omega)$  is zero by definition. The phase of the second term in Eq. (C.25) varies linearly with frequency—thus, we have

$$\Phi(\omega) = -\omega\tau \quad (\text{C.26})$$

Consequently, the obtained FIR filter has linear phase, and its frequency response is denoted as  $H_{lph}(j\omega)$  in Eq. (C.25). Linear-phase filters are the most important applications of the FIR filter. Zero-phase filters can be realized when the signal to be filtered is first recorded and filtered thereafter. As the filter algorithm of Eq. (C.23) demonstrates, the filtered output sample at time  $t = nT$  must then be calculated from a number of samples, which have occurred prior to  $t$ , and from a number of samples, which will occur only later. This becomes possible now because they are already recorded. We can conclude this introduction to FIR filters by stating that the FIR filter is able to approximate an ideal filter to any desired precision with the exception that the filter exhibits a time delay. Two important design techniques for FIR filters are in common use today: (1) the *window technique* and (2) the *Parks-McClellan algorithm*, also called the *Remez algorithm*. When the window technique is used, the filter is effectively designed in the time domain—in other words, its impulse response is derived from the impulse response of the fictive ideal filter. When using the Parks-McClellan algorithm, however, the filter is designed directly in the frequency domain. The term *frequency sampling* was formerly used for this approach.

## Window-FIR filters

When using the window method, the FIR filter is designed in the time domain. The design procedure is illustrated by Fig. C.10. It starts by defining the frequency response of an “ideal” analog filter, which means we set a goal that should be approximated within a given error specification. The “ideal” frequency response is plotted in Fig. C.10a. Next, the impulse response  $h(t)$  of the ideal filter is calculated (Fig. C.10b). To get a finite impulse response, a portion of  $h(t)$  is cut out using the window function  $w(t)$  shown in Fig. C.10c. In this example, the simplest window is used: the *rectangular window*. The finite impulse response  $h^*(t)$  of the FIR filter is given by the multiplication of  $h(t)$  and  $w(t)$  and is plotted in Fig. C.10d. To get a filter that is capable of working in real time, the impulse response is delayed by half its duration, so the first nonzero sample of  $h^*(t)$  starts at  $t = 0$ .

Two parameters of the window have still to be determined: (1) the sampling frequency  $f_s = 1/T$  and (2) the filter length  $N$  (number of taps). The sampling frequency must be chosen to be large enough to avoid aliasing; it must be larger than twice the corner frequency  $f_0$  of the ideal filter. The effect of filter length is discussed in the following. To see its impact, we must calculate the frequency response  $H^*(j\omega)$  of the FIR filter. As the impulse response of the FIR filter is obtained by the multiplication of  $h(t)$  with  $w(t)$ , its frequency response is obtained

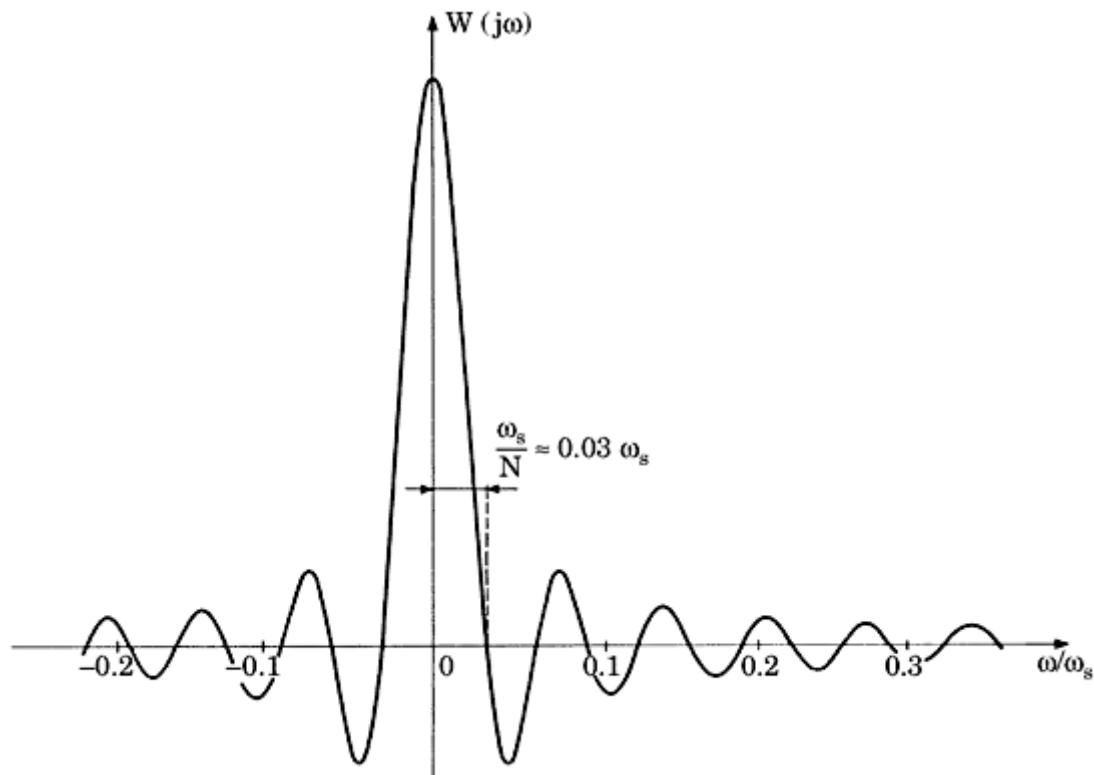
by convolving the spectra  $H(j\omega)$  and  $W(j\omega)$ . The spectrum of the rectangular window can be shown to be<sup>12,13,14</sup>

$$W(j\omega) = \frac{\text{sinc}(\omega NT/2)}{\text{sinc}(\omega T/2)} \quad (\text{C.27})$$

hence is periodic on the frequency scale with the period  $\omega_s = 2\pi/T$ . [Figure C.11](#) shows a portion of the spectrum  $W(j\omega)$ . It consists of a main lobe whose width is

$$\Delta\omega_w = \frac{2\omega_s}{N} \quad (\text{C.28})$$

hence, it is inversely proportional to filter length  $N$ . The amplitudes of the side lobes decay slowly with increasing frequency—meaning they are proportional to  $1/f$ . For an ideal lowpass filter, the result of the convolution  $H(j\omega)*W(j\omega)$  appears in [Fig. C.12](#). In the example shown, the sampling frequency has been chosen to be four times the corner frequency of the filter. The filter length has arbitrarily been chosen as  $N = 31$ . [Figure C.12a](#) shows the frequency response  $H^*(j\omega)$  with logarithmic amplitude scale, and [Fig. C.12b](#) with linear amplitude scale. (For calculation of  $H^*(j\omega)$ , it was assumed the impulse response  $h^*(t)$  is an even function of time—in other words, that the FIR filter is a zero-phase filter. When the FIR filter is required to operate in real time, its amplitude response would simply be the absolute value of the curve shown in [Fig. C.12a](#)).

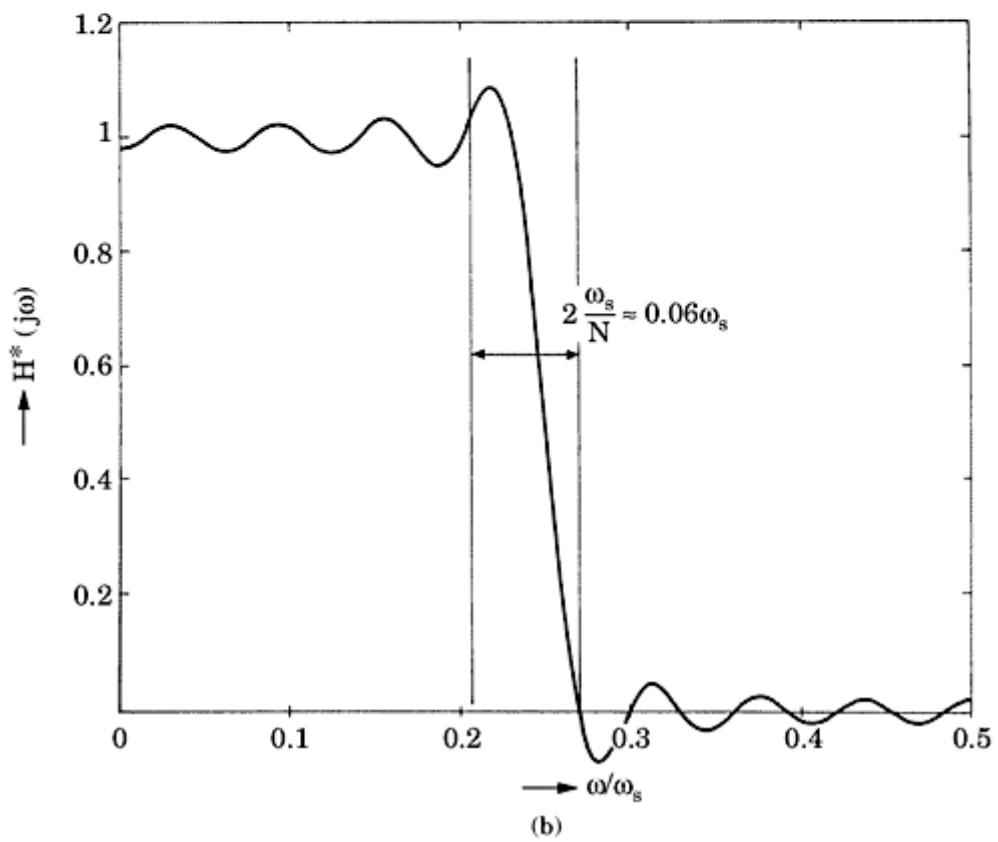
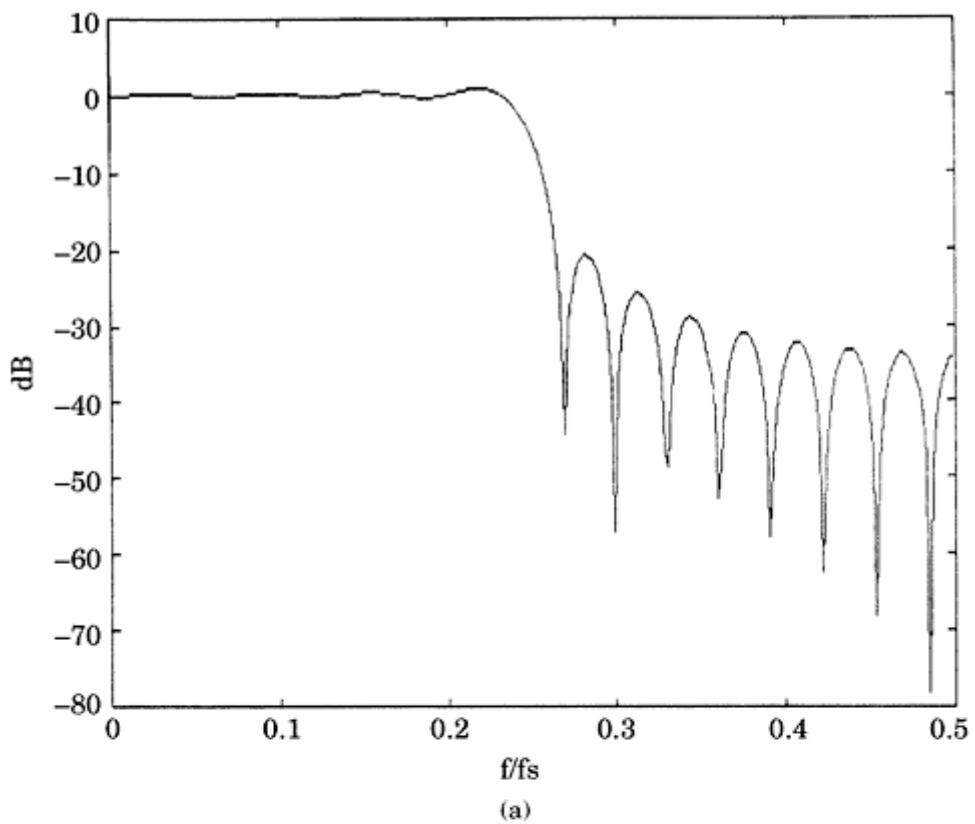


**Figure C.11** Spectrum  $W(j\omega)$  of the rectangular window  $w(t)$  in [Fig. C.10c](#). Only a portion of  $W(j\omega)$  is shown here. The function is periodic in  $\omega$ .

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.





**Figure C.12** The frequency response  $H^*(j\omega)$  of the FIR filter is obtained by the complex convolution  $H(j\omega)^*W(j\omega)$ . (a) The frequency response  $H^*(j\omega)$  on a logarithmic scale. (b) The linear plot of  $H^*(j\omega)$ .

---

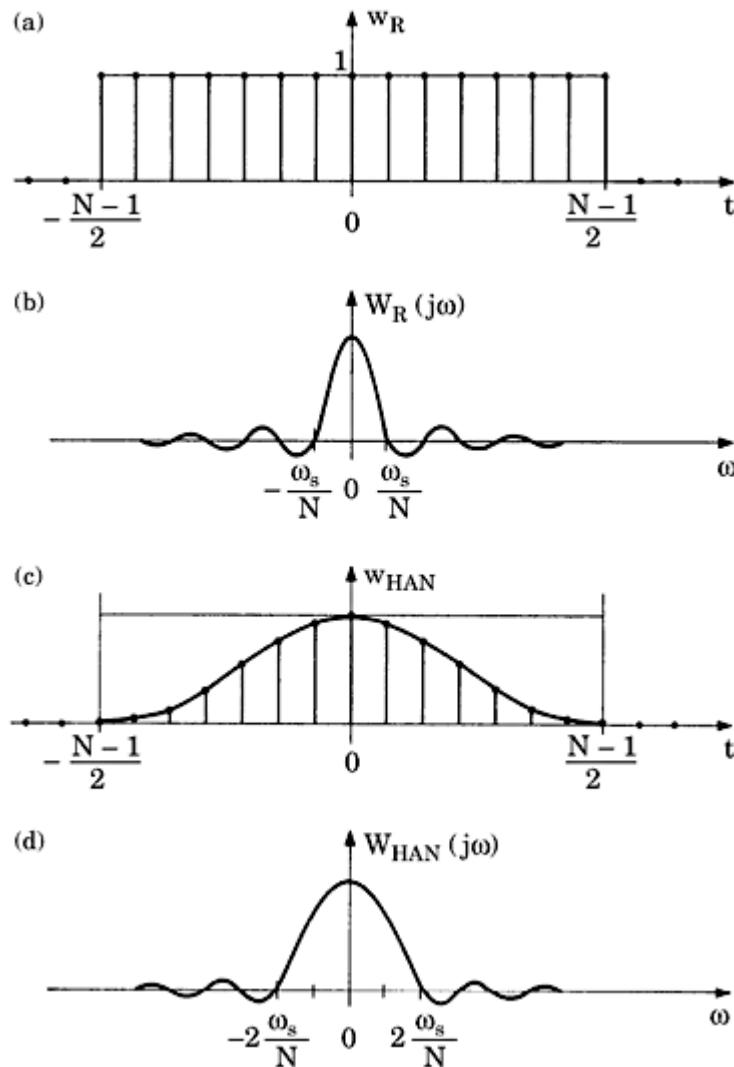
Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

Two effects of the convolution are easily recognized in [Fig. C.12a](#): (1) the steep transition from passband to stopband is widened by an amount that equals the width of the main lobe of  $W(j\omega)$  and is given by [Eq. \(C.28\)](#). The transition width can be made arbitrarily narrow by increasing the filter length  $N$ , but we should keep in mind that increasing  $N$  means also increasing the number of computations to be made in every sampling interval. (2) Owing to convolution, ripple is superimposed to the frequency response of the FIR filter. The ripple amplitude is not constant, moreover. Starting with  $f = 0$ , the ripple in the passband increases with frequency and reaches a maximum near the corner frequency. In the stopband, the ripple amplitude decays only slowly with frequency. The impact of ripple is better recognized from the semilogarithmic representation in [Fig. C.12a](#). Each ripple maximum on the linear amplitude scale corresponds to a damping minimum on the logarithmic scale. The first damping minimum (in other words, the first peak right from the transition in [Fig. C.12a](#)) is a poor  $-21$  dB. At the Nyquist frequency, the damping minima are slightly larger but do not exceed the very modest value of  $-35$  dB. Unfortunately, increasing the filter length does not improve the size of the damping minima. To get improved filter performance in the stopband, we must utilize alternate window functions. Windows must be found, therefore, whose spectrum decays faster with increasing frequency. A great many different windows are known—for example, the Hanning (also called von Hann), Hamming, Bartlett, Blackman, Kaiser, and flattop windows.<sup>[12,13,14,19](#)</sup> Each of these windows has its particular pros and cons.

To save space, we just demonstrate the application of the simplest of these, the Hanning window ([Fig. C.13](#)). To compare it with the known rectangular window, [Fig. C.13a](#) shows once more the function of the rectangular window, which is now denoted  $w_R$ . [Figure C.13b](#) shows again its spectrum  $W_R(j\omega)$ . In [Fig. C.13c](#), the Hanning window is plotted.

Its envelope is simply a cosine function; for this reason, the Hanning window is often referred to as a *raised cosine window*. [Figure C.13d](#) shows the spectrum  $W_{HAN}(j\omega)$  of the Hanning window.

When comparing it with the spectrum of the rectangular window, we note that the main lobe of the Hanning window is twice that of the rectangular. This is clearly a drawback, since it widens the transition region (refer to [Fig. C.12](#)) by a factor of 2. The advantage of the Hanning window is the fact that the side lobes (refer to [Fig. C.13d](#)) decay much faster with increasing frequency—thus, with  $1/f^3$ . When the same FIR lowpass filter as above is realized using the Hanning window with filter length  $N = 31$ , the frequency response shown in [Fig. C.14](#) is obtained. It is easily recognized that the transition has become wider, but the first damping minimum is now at  $-44$  dB (instead of  $-21$  dB for the rectangular window). Still better performance in the stopband is obtained by making use of more sophisticated windows; the Kaiser window is considered the “best” available window because it provides the largest damping minima with the shortest filter lengths.<sup>[12](#)</sup> A common drawback of all window FIR filters is the fact, however, that the ripple in the passband and in the stopband is not constant over frequency.



**Figure C.13** A comparison of different window functions and their spectra. (a) The rectangular window  $w_R(t)$ . (b) The spectrum  $W_R(j\omega)$  of the rectangular window. (c) The Hanning window  $w_{HAN}(t)$ . (d) The spectrum  $W_{HAN}(j\omega)$  of the Hanning window.

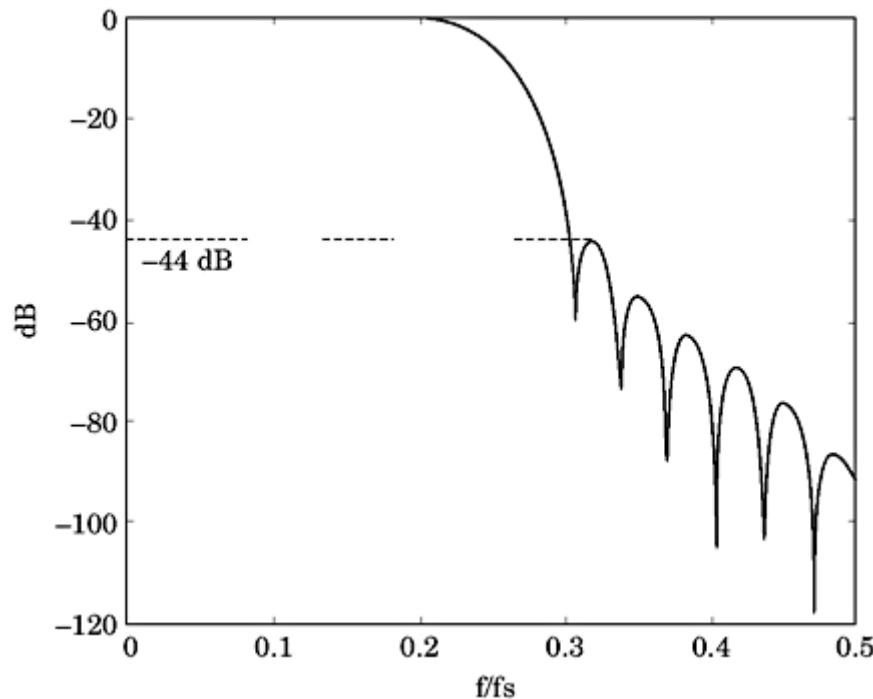
The FIR filters designed with the Parks-McClellan algorithm show constant ripple in both the stopband and passband—hence, they are often called *equiripple filters*. Moreover, the ripple amplitudes can be specified independently for the pass- and stopbands. The Parks-McClellan approach is explained in the next section.

### Designing FIR filters with the Parks-McClellan algorithm

The Parks-McClellan algorithm enables us to design FIR filters that have constant ripple in the passband and in the stopband. The method furthermore allows design of multiband filters, differentiators, and Hilbert transformers. A multiband filter is a filter having more than one passband and stopband. We could think, for example, of a filter passing the frequencies in the range

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.



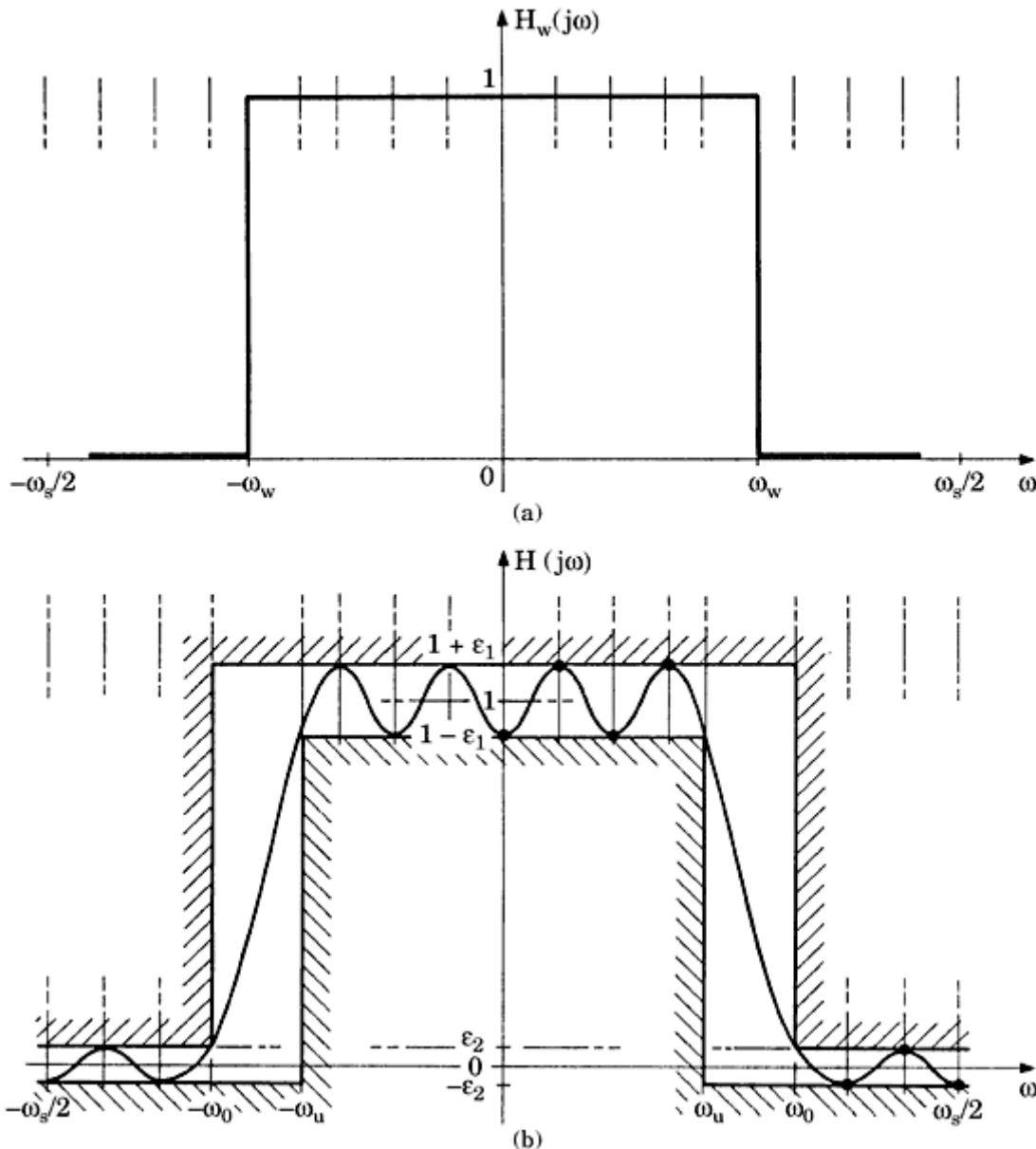
**Figure C.14** The frequency response of a lowpass FIR filter that uses a Hanning window.

of 0 to 1 kHz, rejecting frequencies in the band from 1 to 4 kHz, passing frequencies from 4 to 6 kHz, and so on. When used to design a differentiator, the frequency response of a bandlimited ideal differentiator<sup>12</sup> can be approximated to any desired degree of precision, of course, to the account of filter length. The same holds true for the Hilbert transformer,<sup>13</sup> which is a filter having a gain of 1 and a constant phase shift of  $90^\circ$  over the whole frequency range.

To save space, we concentrate on FIR filters with only one passband, such as the low-pass filter. [Figure C.15](#) illustrates the design procedure. It starts again by plotting the ideal frequency response, which has to be approximated (see [Fig. C.15a](#)). Next, transition width and ripple amplitude are specified by the boundary plot of [Fig. C.15b](#). The FIR filter is required to have constant ripple  $\varepsilon_1$  in the passband and constant ripple  $\varepsilon_2$  in the stopband. The actual frequency must lie entirely within the boundaries, as indicated in the drawing. The (unweighted) error function of the FIR filter is defined to be the difference between ideal and actual frequency response. Consequently, the (unweighted) error function has maxima and minima of  $\pm\varepsilon_1$  in the passband, and  $\pm\varepsilon_2$  in the stopband. Normally, it is desired that  $\varepsilon_2$  be much smaller than  $\varepsilon_1$ . If we wish the FIR filter to have minimum damping of 60 dB in the stopband, for example,  $\varepsilon_2$  becomes 0.001. It would not be reasonable now to postulate that the ripple in the passband should also be as small as 0.001; it would probably be sufficient to have a passband ripple on the order of 0.1, which is 100 times the allowed stopband ripple. We now introduce a weighting function  $W(j\omega)$  such that the *weighted* ripples in passband and stopband have the same size. The weighting function  $W(j\omega)$  is shown in [Fig. C.15c](#). In our example, the allowed ripple in the stopband is half the ripple in the passband; hence, the weighting function is

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

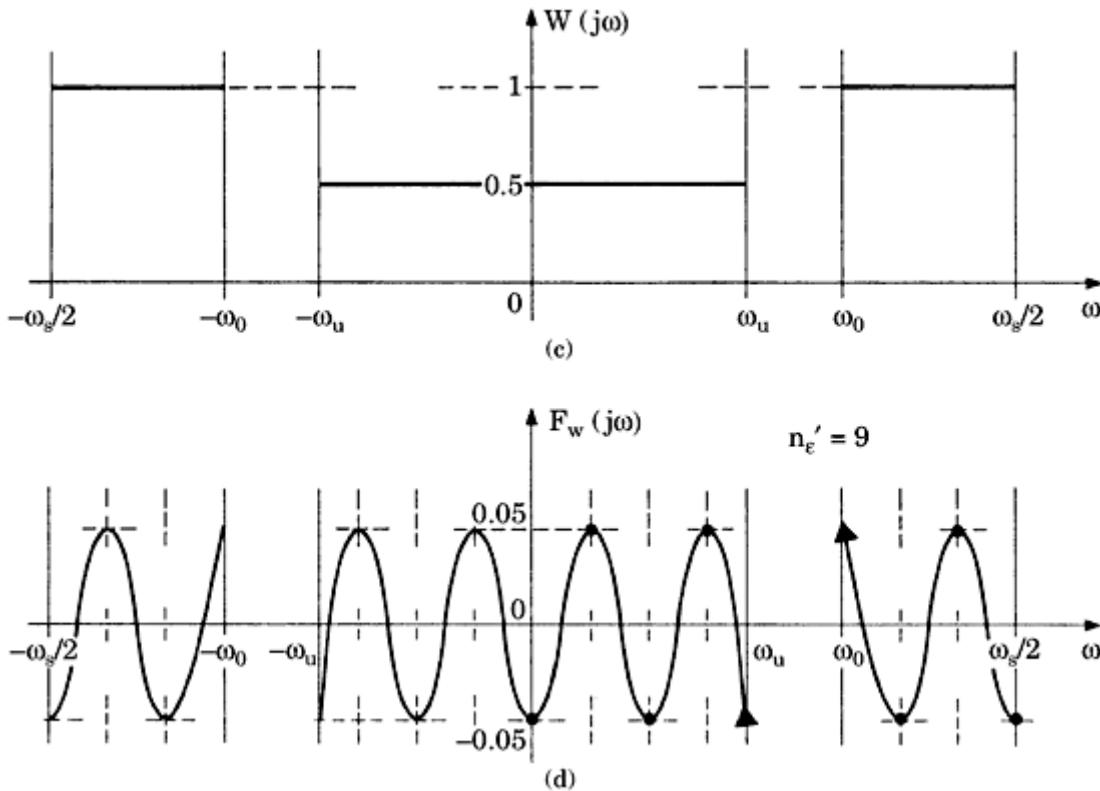


**Figure C.15** This figure illustrates the design procedure for the Parks-McClellan algorithm.  
 (a) The frequency response  $H(j\omega)$  of an ideal analog filter is specified. (b) A boundary plot specifies the deviations of the actual filter from the ideal. In this plot, transition width and maximum ripple in the passband and stopband are defined.

chosen as 1 in the stopband and 0.5 in the passband. Finally, Fig. C.15d shows the weighted error function  $F_w(j\omega)$  of the FIR filter. The weighted error function is obtained by multiplying the unweighted error function by  $W(j\omega)$ . The famous *Chebyshev polynomials* are known to behave like the weighted error curve shown in Fig. C.15d.

The Parks-McClellan algorithm is an iterative procedure that tailors a function in the frequency domain such that it comes closer and closer to the desired weighted error function  $F_w(j\omega)$ . It can be shown that for a filter length  $N$ , the number of extrema (meaning, minima

plus maxima) of the weighted error function in the frequency range 0 to  $f_n$  (Nyquist frequency) is around  $N/2$ . (The exact figure depends somewhat on the type of FIR filter to be realized; we will not go



**Figure C.15 (Continued)** (c) Because different amounts of ripple are tolerated in pass-band and stopband, a weighting function  $W(j\omega)$  is introduced. (d) The weighted error curve  $F_w(j\omega)$  is given by the product of absolute error (from Fig. C.15b) and weighting function  $W(j\omega)$  (from Fig. C.15c).

into details here.<sup>13</sup>) The Parks-McClellan algorithm starts with an “initial guess” of the locations of the extrema of  $F_w(j\omega)$  (see the filled dots in Fig. C.15d). Because it is very difficult to predict these locations accurately, the Parks-McClellan algorithm determines in the first pass a polynomial that passes through the filled dots; to simplify the mathematics, a *Lagrange polynomial* is used.<sup>13</sup> (When  $N$  points of a curve are specified, it is relatively simple to determine a polynomial of degree  $N - 1$  that goes through these points exactly; this kind of polynomial is called a Lagrange polynomial and is often used for the interpolation of functions.) If the initial guess would have been perfect, the extrema of the fitted Lagrange polynomial would be exactly there, where they were supposed to be. Probably the true extrema of the Lagrange polynomial will be at different locations, however. A maxima/minima search is started now, which determines the locations where the extrema really are. This set of new values is used now as an improved initial guess. In the next run, the Parks-McClellan algorithm fits another Lagrange polynomial through the new points. It performs as many passes as needed until the extrema found in the  $K$ th pass do not markedly deviate from the extrema found in the  $K - 1$ th pass. Having found a sufficiently good fit to the weighted error function  $F_w(j\omega)$ , the Parks-McClellan algorithm now calculates the unweighted error function and finally the actual frequency response  $H^*(j\omega)$  of the FIR filter. Actually, a sampled

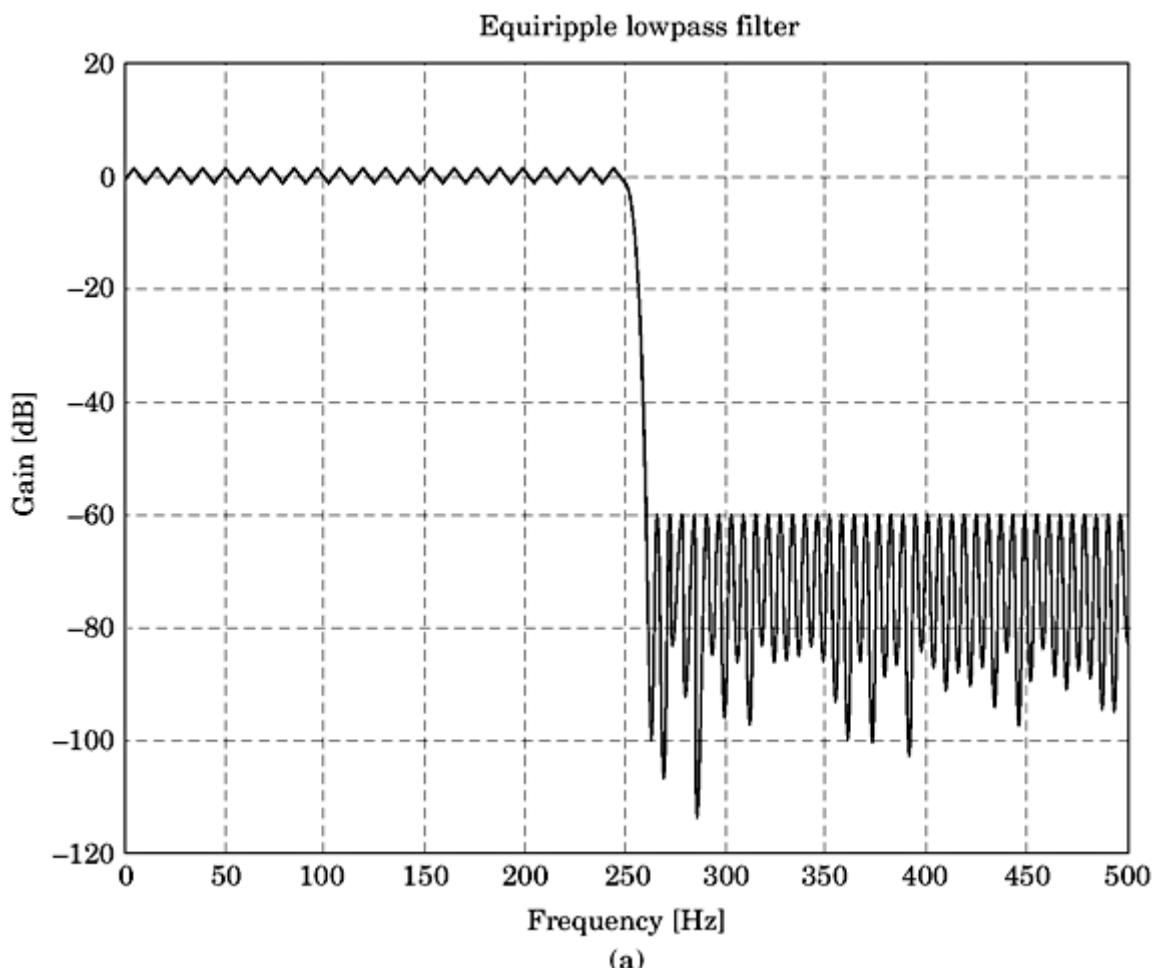
---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

version of  $H^*(j\omega)$  is known at this time. Using the IFFT (inverse FFT), the actual impulse response  $h^*(t)$  is calculated. This yields the required FIR filter coefficients  $h(0), h(1), \dots, h(N-1)$ , where  $N$  is the filter length. [Figure C.16](#) is an example of an FIR lowpass filter of length  $N = 180$ , which has been designed by the signal processing toolbox of Matlab.<sup>35</sup> This filter has the following specifications:

- Sampling frequency  $f_s = 1000$  Hz
- Passband 0 to 250 Hz
- Stopband 260 to 500 Hz
- Maximum passband ripple  $\varepsilon_1 = 0.1$  (corresponding to approximately 0.8 dB)
- Maximum stopband ripple  $\varepsilon_2 = 0.001$  (corresponding to  $-60$  dB)

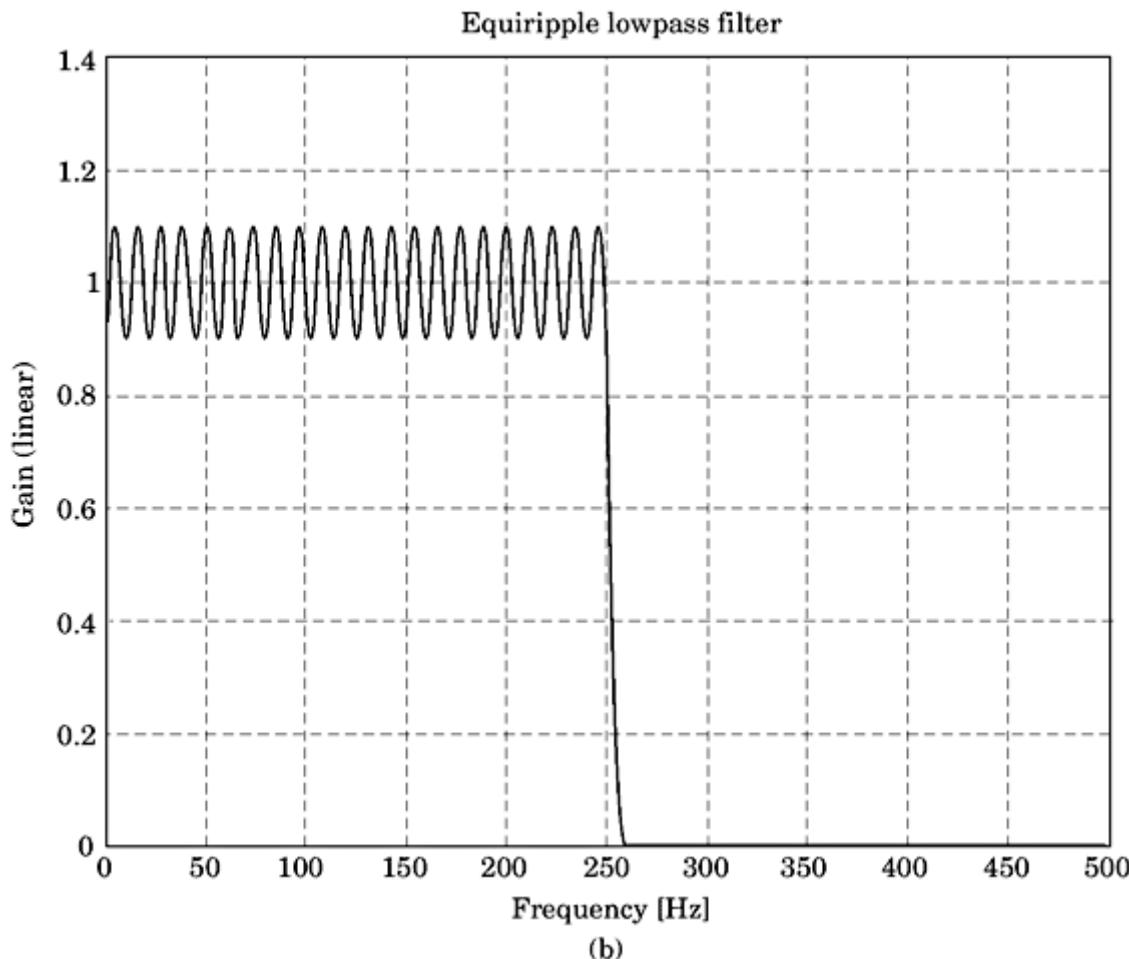
Before computing the filter coefficients, function *remezord* is used to determine the filter length required to meet the design specifications. *remezord* specifies a filter length of  $N = 180$ . This may seem quite high, but we postulated a



**Figure C.16** An equiripple lowpass filter designed with Matlab.<sup>35</sup> The maximum ripple in the passband is 0.1 (0.8 dB), the maximum stopband ripple is 0.001 ( $-60$  dB), and the transition width is 10 Hz. To meet these specifications, a filter length  $N = 180$  is required. (a) Gain versus frequency on a semilogarithmic scale.

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.



**Figure C.16 (Continued) (b)** Gain versus frequency on a linear scale.

very low stopband ripple and a rather narrow transition bandwidth (that is, 10 Hz). [Figure C.16a](#) shows the filter gain versus frequency on a semilogarithmic scale. We clearly see that the stopband ripple is at a constant  $-60$  dB level. To check the passband ripple, it is more convenient to display the gain versus frequency curve on a linear scale ([Fig. C.16b](#)). Also, here we note that the pass-band ripple exactly meets the design goal ( $\pm 0.1$ ).

The Parks-McClellan algorithm is an extremely computation-intensive procedure. Only computer programs are capable of performing such an approach in reasonable time. Fortunately, many software tools are available that enable the design of any kind of digital filters—that is, IIR filters, FIR filters with the window method, and also with the Parks-McClellan algorithm.[25,35](#) When using such a software tool, the designer will first enter the error specifications of the FIR filter. Knowing ripple amplitudes and transition width(s), most programs estimate the filter length  $N$  required to meet the design goals. The filter length depends to a large extent on the required transition width. The narrower the transition, the longer the filter will be.

Another problem also deserves attention: the number of bits that are necessary to represent the filter coefficients in the hardware platform, which is used to perform the filtering function. The mentioned computer programs calculate

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

the filter coefficients with the precision of floating-point numbers—that is, usually to a precision of 32, 64, or even 80 bits. (Note that these numbers include the bits used for mantissa and exponent.) In many cases, the user will try to implement the filter algorithm with fixed-point arithmetic, which may use a 16-bit (or even an eight-bit) format. The actual filter coefficients must therefore be rounded. Because of rounding, the frequency response can be corrupted, however. The filter performance in the stopband will most likely be deteriorated. To check numerical resolution, most filter design tools offer an option to round the filter coefficients and to recalculate the frequency response with the truncated coefficients.

## Measuring PLL Parameters

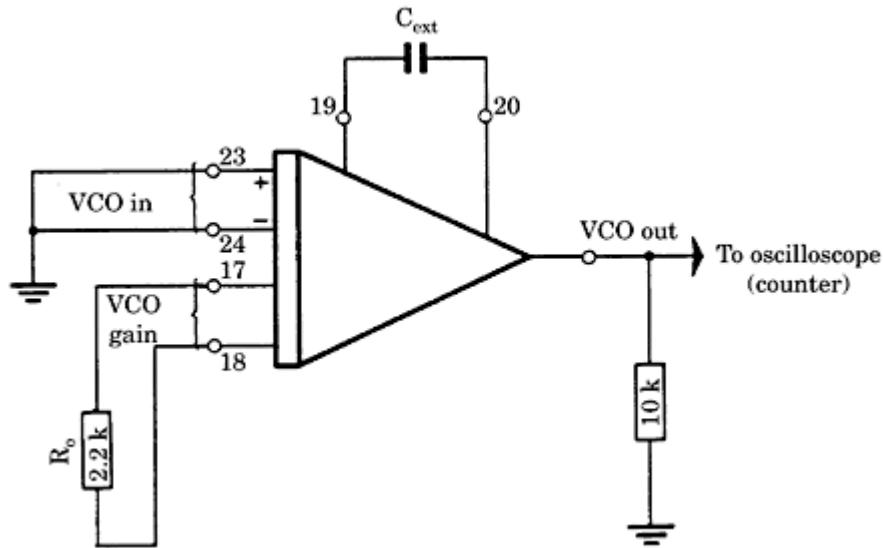
This chapter deals with the measurement of PLL parameters. When an ADPLL is used, nothing has to be measured, because all parameters of the circuit are multipliers for clock frequencies and divider ratios of counters. In the case of the DPLL, the parameters are normally well specified in the data sheets. The phase-detector gain, for example, depends uniquely on the supply voltage in most cases. For a DPLL built from CMOS technology, the levels of the phase-detector output signal come close to the supply rails, so the phase-detector gain  $K_d$  is approximately  $U_B/\pi$  for the EXOR,  $U_B/2\pi$  for the JK-flipflop, and  $U_B/4\pi$  for the PFD. Moreover, the VCO characteristics are well specified on the data sheets of the 74 HC/HCT4046, 74 HC/HCT7046, and 74 HC/HCT9046 ICs, so there is sufficient information to calculate the values of the external components of the DPLL system. The situation is different in the case of LPLLs, where some parameters are poorly specified for some products.

Many LPLL ICs have a large supply voltage range, and some of the PLL parameters can vary with it. Furthermore, the phase-detector gain depends on the level of the reference signal.

It is therefore advantageous if the users are able to measure these parameters themselves. It will be shown in this section that the relevant parameters such as  $K_0$ ,  $K_d$ ,  $\omega_n$ ,  $\zeta$ , and many others are easily measured with the standard equipment available even in a hobbyist's lab, say, an oscilloscope and a waveform generator. For the following measurements, a multifunction integrated circuit of type XR-S200 (EXAR) has been arbitrarily selected as the DUT.

### Measurement of Center Frequency $f_0$

Only the VCO portion of the DUT is used for this measurement (see Fig. D.1). The two pins of the symmetrical VCO input are grounded. Consequently, the VCO will oscillate at the center frequency  $f_0$ . The center frequency is now easily measured with an oscilloscope (Fig. D.2a). If the values  $C_{\text{ext}} = 82 \text{ nF}$  and  $R_0 = 2.2 \text{ k}\Omega$  are chosen for the external components, the VCO oscillates at a center frequency  $f_0$  of 6.54 kHz.



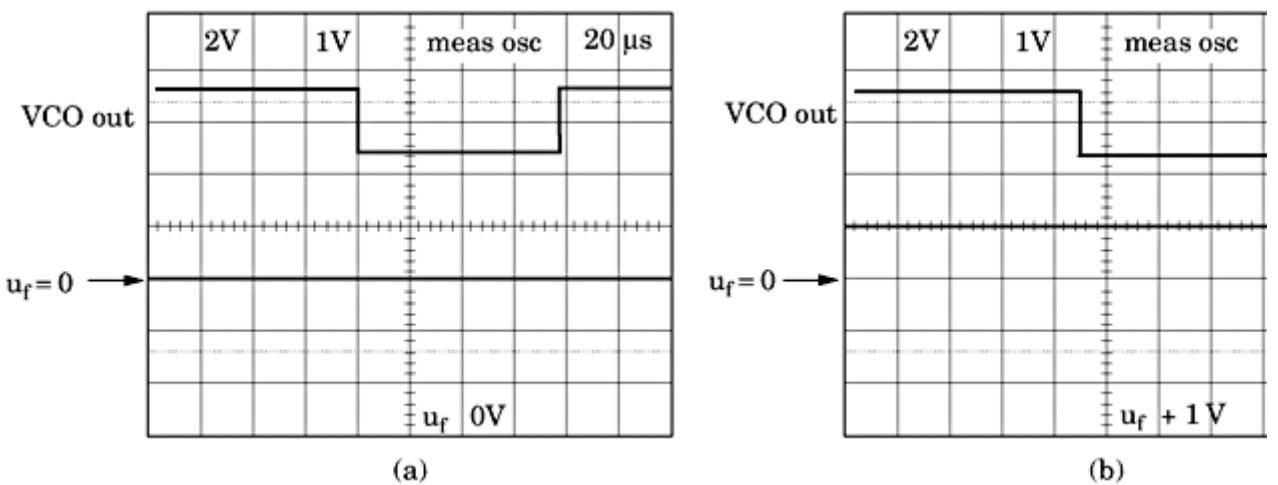
**Figure D.1** The test circuit for the measurement of the center frequency  $\omega_0$  and the VCO gain  $K_0$ .

### Measurement of VCO Gain $K_0$

The same test circuit (see Fig. D.1) can be used to measure the VCO gain  $K_0$ . One of the VCO inputs (VCO in) stays grounded; a variable DC voltage is applied to the other. This signal corresponds to the loop filter output signal  $u_f$ . By definition, the VCO gain  $K_0$  is equal to the variation of VCO angular frequency  $\Delta\omega_0$  related to a variation of the  $u_f$  signal by  $\Delta u_f = 1 \text{ V}$ . As shown in Fig. D.2b, the VCO frequency falls to 5.78 kHz for  $\Delta u_f = 1 \text{ V}$ , which corresponds to a variation of 0.76 kHz.

The sign of the frequency variation is irrelevant here; it would have been positive if the VCO input pins had been connected with inverted polarity. For the VCO gain  $K_0$ , we now obtain

$$K_0 = \frac{\Delta\omega_0}{\Delta u_f} = \frac{2\pi \cdot 0.76 \cdot 10^3}{1} = 4.78 \cdot 10^3 \text{ rad s}^{-1} \text{ V}^{-1}$$



**Figure D.2** (a) Measurement of  $\omega_0$ ,  $u_f = 0$  V. (b) Measurement of  $K_0$ ,  $u_f = 1$  V.

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
 Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
 Any use is subject to the Terms of Use as given at the website.

The test shows furthermore that  $K_0$  would be larger by a factor of 10 if  $\omega_0$  had been chosen to be larger by a factor of 10. Hence, for this device,  $K_0$  varies in proportion to  $\omega_0$ . When  $R_0 = 2.2 \text{ k}\Omega$  is chosen,  $K_0$  is given by the general equation

$$K_0 = 0.73f_0 \text{ rad s}^{-1} \text{ V}^{-1} \quad (\text{D.1})$$

where  $f_0$  is in Hertz. For example, for  $f_0 = 1 \text{ kHz}$ , we find  $K_0 = 730 \text{ rad s}^{-1} \text{ V}^{-1}$ .

## Measurement of Phase-Detector Gain $K_d$

The test circuit of [Fig. D.3](#) is used for the measurement of  $K_d$ . The PD of the XR-S200 is realized in the form of an operational multiplier. In the configuration shown, the PD and the VCO are used. The VCO output signal is coupled capacitively to one of the PD inputs. A variable DC level is applied to the other input.

The measurement procedure is explained by the waveforms in [Fig. D.4](#). In [Fig. D.4a](#), the signal applied to phase-detector input 1 (PD in #1) is a sine

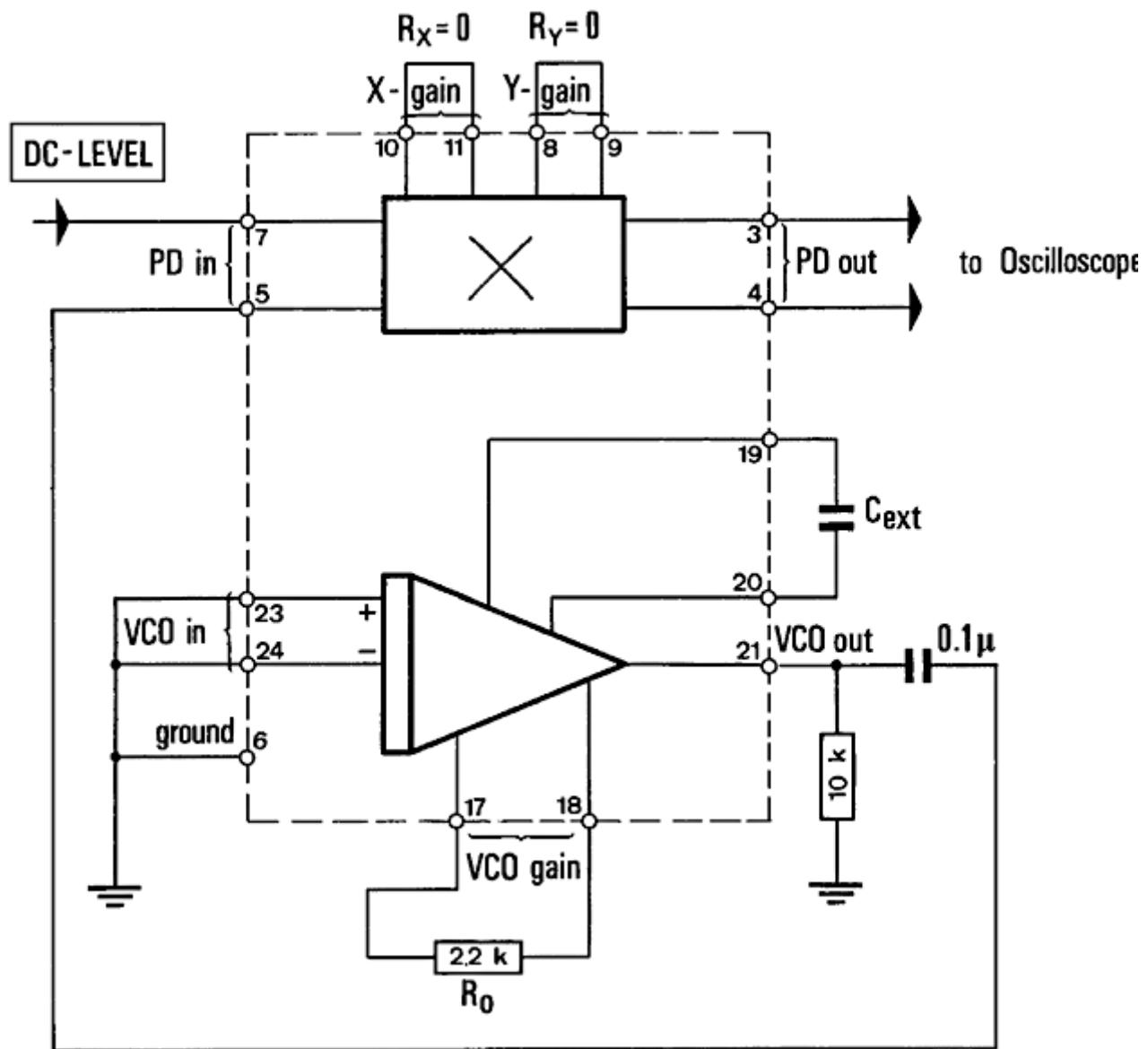
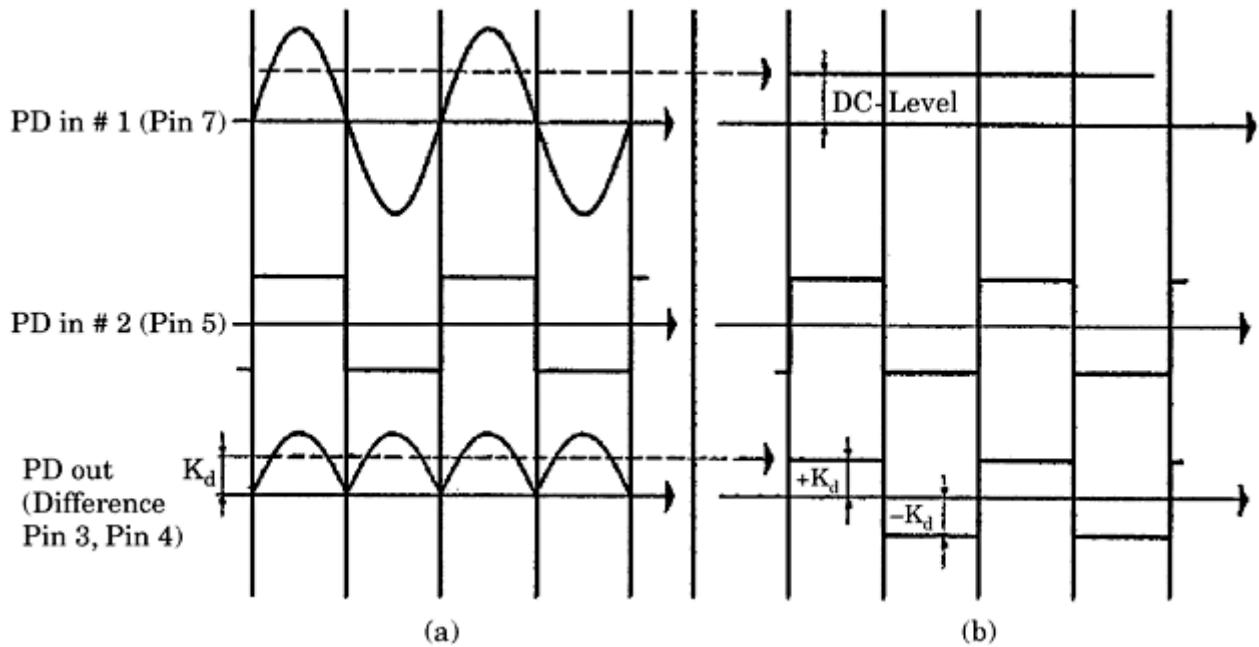


Figure D.3 A test circuit for the measurement of phase-detector gain  $K_d$ .



**Figure D.4** Waveforms of the phase-detector output signal  $u_d$  for different signals applied to input 1. (a) For a sine-wave signal. (b) For a DC level.

wave, and the signal applied to phase-detector input 2 (PD in #2) is a square wave (usually the VCO output signal). It is furthermore assumed that both signals are in phase. Consequently, the phase error  $\theta_e$  is  $90^\circ$ . The phase-detector output signal (PD out) is a full-wave rectified sine signal. Its average value is given by

$$\overline{u_d} = K_d \sin 90^\circ = K_d$$

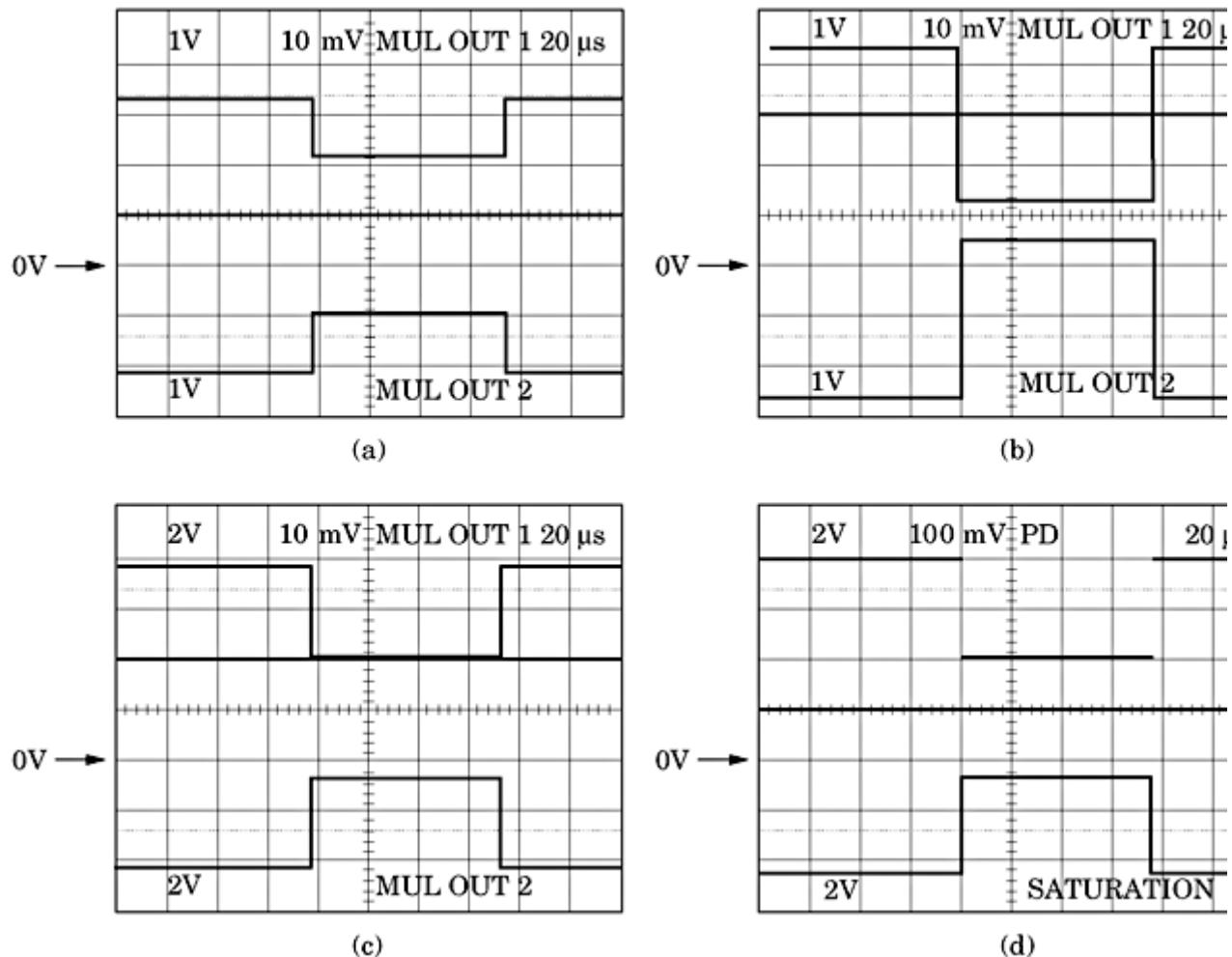
Thus, the measured output signal is identical with  $K_d$ .

If a DC voltage is applied to PD in #1, whose level is equal to the average value of the previously applied sine signal, the output signal of the phase detector becomes a square wave having a peak amplitude  $K_d$ . In most data sheets,  $K_d$  is specified as a function of the reference signal level; a sine signal is usually chosen for the reference signal, and the signal level is usually given as an rms value. In Fig. D.4b, we see that the rms value of the sine signal is 1.11 times its average value. (For a sine signal of peak amplitude 1, the rms value is  $1/\sqrt{2}$ , and the average value [linear average] is  $2/\pi$ .) Consequently, to measure  $K_d$  for a reference signal level of 1 V rms, we must apply a DC level of  $1/1.11 \approx 0.9$  V to PD in #1 and then simply measure the peak value of the output signal.

The phase-detector gain of the DUT in Fig. D.3 was measured at four different levels of the reference signal: 10, 30, 40, and 100 mV rms. The measured signals are displayed in Fig. D.5. The four oscilloscopes show the results for  $u_1 = 10, 30, 40$ , and 100 mV rms, respectively. As Fig. D.3 demonstrates, this PD has a symmetrical output.

Three signals are displayed in each oscilloscope: the top trace shows multiplier output 1 (MUL OUT 1), the center trace the reference signal (DC), and the bottom trace multiplier

output 2 (MUL OUT 2).



**Figure D.5** Waveforms measured with the test circuit of Fig. D.3 for different DC levels applied to input 1. (a) 10 mV. (b) 30 mV. (c) 40 mV. (d) 100 mV.

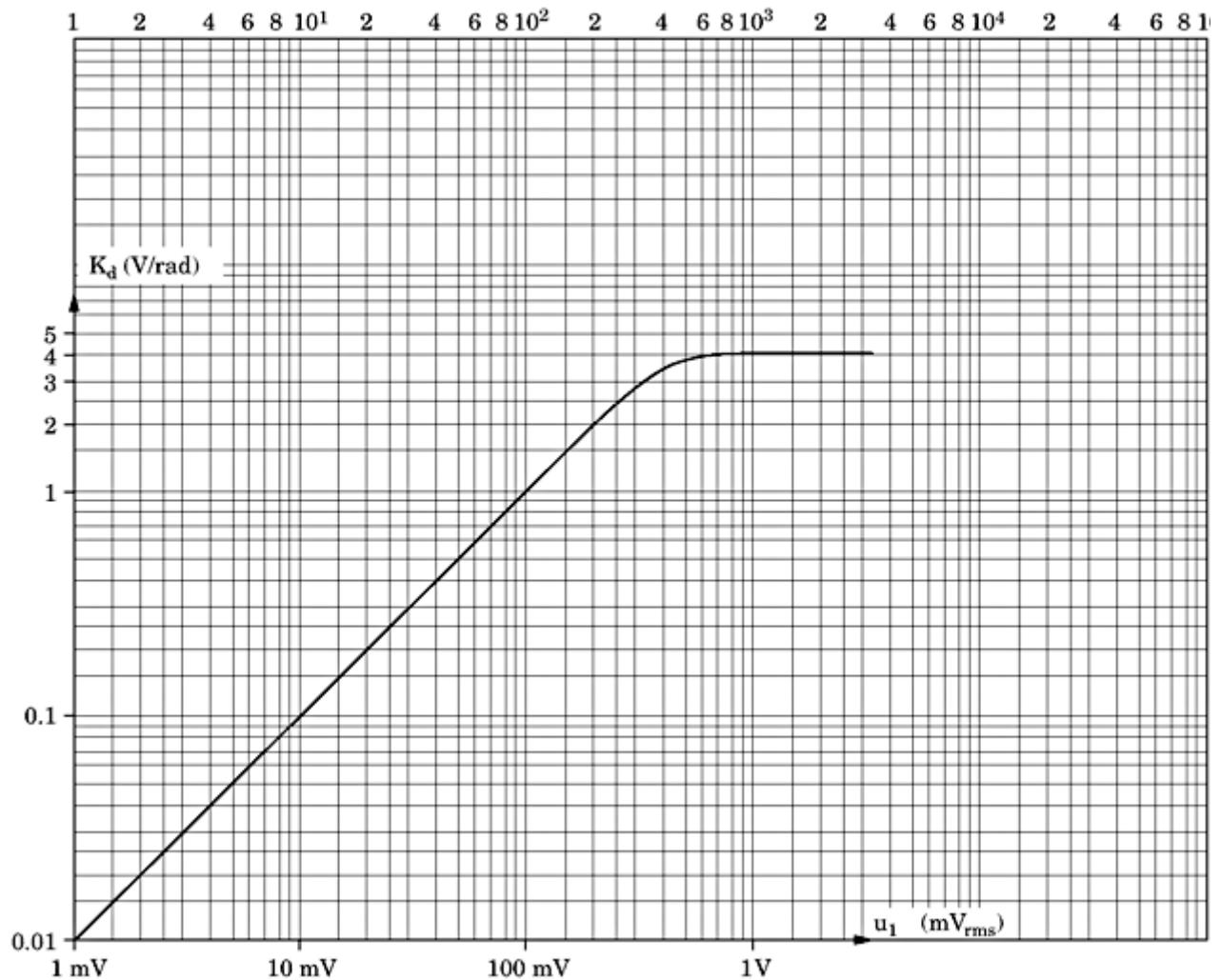
Because the output of the PD is a symmetrical signal, the phase-detector gain is *twice* the measured peak amplitude of the square wave. This means that the PD output signal (shown in the bottom trace of Fig. D.4) must be measured differentially *between pins 3 and 4*. The phase-detector gain is then the amplitude of the square wave measured from the centerline, as indicated in Fig. D.4.

The measured  $K_d$  values are plotted against the reference signal level in Fig. D.6. It is clearly observed that the PD becomes saturated at signal levels above 400 mV rms.

### Measurement of Hold Range $\Delta\omega_H$ and Pull-in Range $\Delta\omega_P$

To measure these parameters, a full PLL circuit must be built. This is easily done by adding a loop filter to the test circuit of Fig. D.3 and by closing the loop. The new test circuit of Fig. D.7 is then obtained. This PLL does not use a down scaler, hence we have  $N = 1$  ( $N$  = scaling factor). A passive loop filter (Fig. 2.17a) is chosen for simplicity; it consists of two on-chip

resistors,  $R_1 = 6 \text{ k}\Omega$  each, and



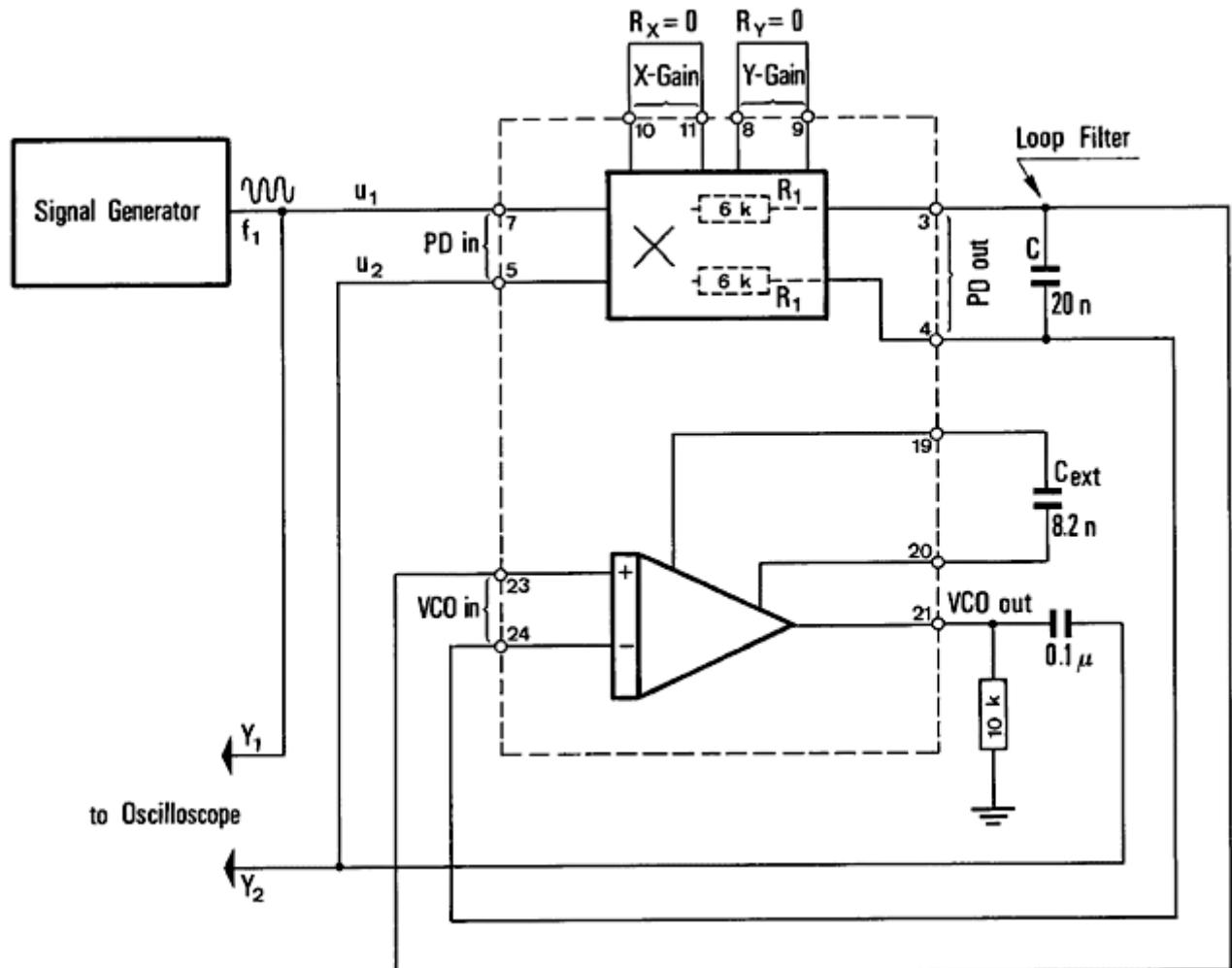
**Figure D.6** A plot of  $K_d$  against the input voltage level in millivolts rms.

the external capacitor  $C$ . (Note that resistor  $R_2$  is set to 0 here, although this is a bad choice for loop stability.) A signal generator is applied to the reference input (pin 7) of the DUT.

Both reference signal  $u_1$  and VCO output signal  $u_2$  are displayed versus time on an oscilloscope. The oscilloscope is triggered on  $u_2$ . The frequency of the signal generator is now varied manually until lock-in is observed (Fig. D.8). In the case of Fig. D.8a, the reference frequency  $f_1$  is far away from the center frequency  $f_0$ , and the system is unlocked. Figure D.8b shows the situation where  $f_1$  has come closer to the center frequency. The PLL is trying to pull in the VCO. Frequency modulation of  $u_2$  is easily observable. If the reference frequency approaches  $f_0$  slightly more, the PLL suddenly locks (see Fig. D.8c).

Determination of the hold range  $\Delta f_H$  and the pull-in range  $\Delta f_P$  is very simple. The hold range is measured by slowly varying the reference frequency  $f_1$  and monitoring the upper and lower values of  $f_1$  where the system unlocks.

The pull-in range  $\Delta f_P$  is determined in a similar way. To get  $\Delta f_P$ , we first set the reference frequency  $f_1$  to approximately the center frequency  $f_0$  and then increase  $f_1$  slowly until the loop locks out. To see where the loop pulls in again,



**Figure D.7** The test circuit for the measurement of hold range  $\Delta\omega_H$  and pull-in range  $\Delta\omega_P$ .

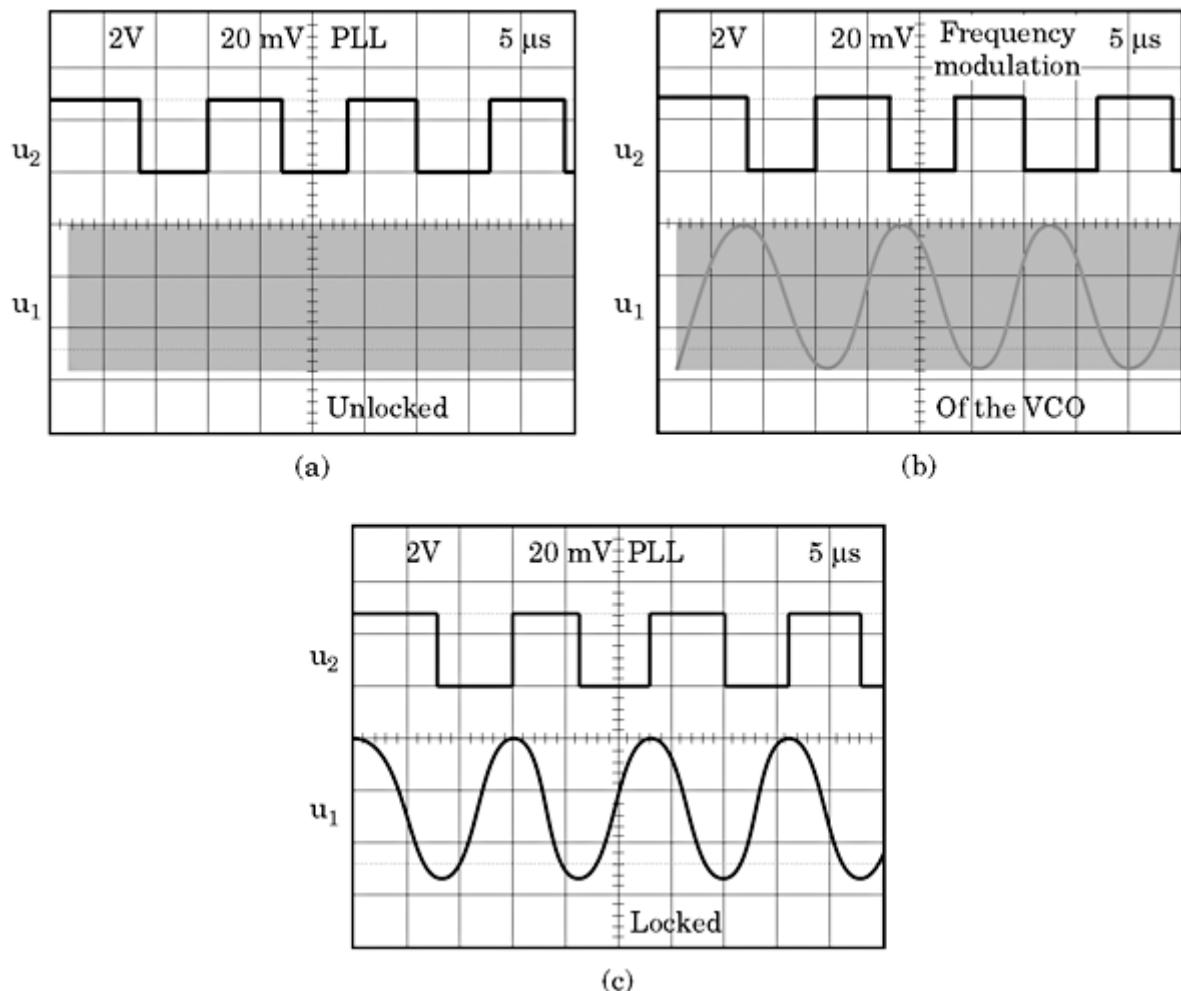
we must reduce the reference frequency *slowly* and monitor the value of  $f_1$  where the loop pulls in. The difference between the value of  $f_1$  and the center frequency  $f_0$  is the pull-in range  $\Delta f_P$ .

### Measurement of Natural Frequency $\omega_n$ , Damping Factor $\zeta$ , and Lock Range $\Delta\omega_L$

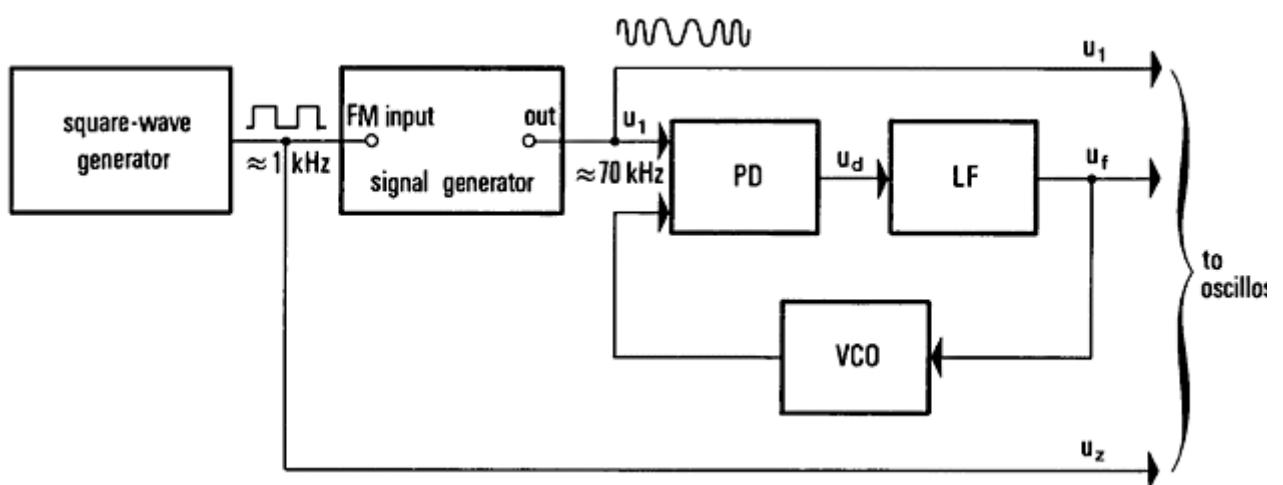
The PLL circuit of Fig. D.7 is used for the following measurements. To measure the natural frequency  $\omega_n$  and the damping factor  $\zeta$  of a PLL, we apply a disturbance to the PLL which forces the system to settle at a different stable state. This is most easily done by modulating the reference frequency with a square-wave signal. The corresponding test circuit is shown in Fig. D.9. The PLL being tested operates at a center frequency of approximately 70 kHz. The frequency of the signal generator is modulated by a square-wave generator. Of course, the

modulating frequency must be chosen to be much smaller than the center frequency—for example, 1 kHz.

If the frequency of the signal generator is abruptly changed, a phase error  $\theta_e$  results. The output signal  $u_f$  of the loop filter can be considered a measure of the average phase error. Thus, the transient response of the PLL can be easily



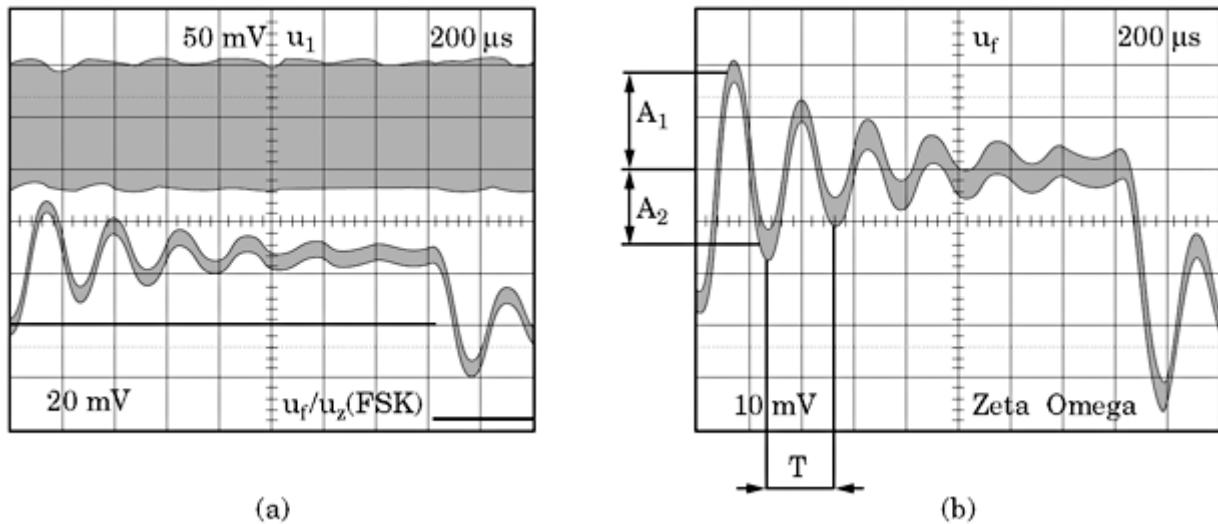
**Figure D.8** Waveforms of signals  $u_1$  and  $u_2$  in the test circuit of Fig. D.7. (a) PLL unlocked. (b) PLL near locking. (c) PLL locked.



**Figure D.9** The test circuit for the measurement of natural frequency  $\omega_n$  and the damping factor  $\zeta$ .

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.



**Figure D.10** Waveforms of the test circuit shown in Fig. D.9. (a) Top trace—input signal  $u_1$ ; center trace—output signal of the loop filter; bottom trace—modulating signal, 1-kHz square wave. (b) Enlarged view of the  $u_f$  waveform shown in (a), center trace.

analyzed by recording on an oscilloscope. This is shown by the waveforms in Fig. D.10a, which displays the signals  $u_1$  (reference signal, top trace),  $u_f$  (center trace), and the 1-kHz square wave (bottom trace). The oscilloscope is triggered on the 1-kHz square wave. Because the reference signal has a much higher frequency than the 1-kHz square wave and is by no means synchronized with the latter, it is displayed as a bar only.

The  $u_f$  signal performs a damped oscillation on every transient of the 1-kHz square wave and settles at a stable level thereafter. Now  $\omega_n$  and  $\zeta$  can be calculated from the waveform of  $u_f$ . Figure D.10b is an enlarged view of this signal;  $\zeta$  can be calculated from the ratio of the amplitudes of two subsequent half-waves  $A_1$  and  $A_2$ . (Any pair of subsequent half-waves may be chosen.) The damping factor is given by

$$\zeta = \frac{\ln(A_1/A_2)}{[\pi^2 + (\ln(A_1/A_2))^2]^{1/2}}$$

The natural frequency  $\omega_n$  is calculated from the period  $T$  of one oscillation in Fig. D.10b according to

$$\omega_n = \frac{2\pi}{T\sqrt{1 - \zeta^2}}$$

Let's now evaluate numerically the waveform of Fig. D.10b. For  $A_1$  and  $A_2$ , we read approximately 1.9 and 1.5 divisions, respectively. Consequently, we obtain

$$\zeta \approx 0.08$$

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

For  $T$ , we find  $T \approx 240 \mu\text{s}$ . Hence,  $\omega_n$  is

$$\omega_n \approx 26.0 \cdot 10^3 \text{ rad s}^{-1}$$

or

$$f_n \approx 4.1 \text{ kHz}$$

To complete this measurement, let's calculate the values of  $\omega_n$  and  $\zeta$  using the theory of the linear PLL [Eq. (3.13)] and check whether our measurements agree with the predicted results. Using Eq. (D.1), we obtain for the VCO gain

$$K_0 = 0.73 \cdot 70 \cdot 10^3 = 51.5 \cdot 10^3 \text{ rad s}^{-1} \text{ V}^{-1}$$

According to Fig. D.10a, the amplitude of the reference signal is about 120 mV peak-to-peak, which corresponds to an rms value of 43 mV. For this signal level, we read a phase-detector gain  $K_d \approx 3.7 \text{ V/rad}$  from Fig. D.6. The two time constants  $\tau_1$  and  $\tau_2$  can be derived from the test circuit in Fig. D.7

$$\begin{aligned}\tau_1 &= 12k\Omega \cdot 20nF = 240 \mu\text{s} \\ \tau_2 &= 0\end{aligned}$$

Using Eq. (2.41), we finally get

$$\omega_n = \left( \frac{K_0 K_d}{N(\tau_1 + \tau_2)} \right)^{1/2} = \left( \frac{51.1 \cdot 10^3 \cdot 3.7}{240 \cdot 10^{-6}} \right)^{1/2} = 28 \cdot 10^3 \text{ rad s}^{-1}$$

$$\zeta = \frac{\omega_n}{2} \left( \tau_2 + \frac{N}{K_0 K_d} \right) = \frac{28 \cdot 10^3}{2 \cdot 51.1 \cdot 10^3 \cdot 3.7} = 0.07$$

This agrees well with the experimental measurements. Note that this experimental method of measuring  $\omega_n$  and  $\zeta$  is applicable only for  $\zeta < 1$ . This requirement is met, however, in most cases. If  $\zeta$  were greater than 1, the transient response would become aperiodic, and it would be impossible to define the values  $A_1$  and  $A_2$  in Fig. D.10a. In the example of Fig. D.10a, the damping factor  $\zeta$  has purposely been chosen to be too small in order to get a marked oscillatory transient. An underdamped system is often obtained when a loop filter without a zero is chosen. To increase  $\zeta$ ,  $\tau_2$  should be chosen  $> 0$ .

Measurement of the lock range  $\Delta\omega_L$  is possible, though not so easy. It can be done using the setup shown in Fig. D.9 which was built to measure the natural frequency and damping factor. The signal generator is still frequency modulated by the square-wave generator, as shown. Assume that the radian center frequency of the PLL under test is  $\omega_0$ . The higher of the two frequencies ( $\omega_{\text{high}}$ ) generated by the signal generator must now be chosen higher than the pull-out frequency—that is,  $\omega_{\text{high}} > \omega_0 + \Delta\omega_P$ . The lower frequency ( $\omega_{\text{low}}$ ) must be chosen

initially to be as high as the higher ( $\omega_{\text{low}} = \omega_{\text{high}}$ ). In this situation,

the amplitude of the square-wave generator will be zero. The test circuit must now be tuned such that  $\omega_{\text{high}}$  always stays the same, but  $\omega_{\text{low}}$  decreases when the square-wave amplitude is made larger. The square-wave amplitude is now continually increased while watching the  $u_f$  signal with the scope, as shown in Fig. D.10a. During the interval where the frequency of the signal generator is high,  $u_f$  will be a “high-frequency” signal, which is only visible as a “smeared” trace. When the lower frequency reaches a value around  $\omega_{\text{low}} \approx \omega_0 + \Delta\omega_L$ , the PLL will lock quickly—in other words, the  $u_f$  signal will settle to a stable value within at most one cycle (oscillation). The lock range  $\Delta\omega_L$  is now given by the difference  $\omega_{\text{low}} - \omega_0$ . For larger values of  $\omega_{\text{low}}$ , a pull-in process will be observed: the  $u_f$  signal also settles to some finite value, but shows up more than one cycle.

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

## References

1. Gardner, Floyd M.: *Phaselock Techniques*, 2d ed., John Wiley and Sons, New York, 1979.
2. Richman, D.: "Color Carrier Reference Phase Synchronization Accuracy in NTSC Color Television," *Proc. IRE*, Vol. 42, January 1954, pp. 106–133.
3. Izawa, K.: *Introduction to Automatic Control*, Elsevier, New York, 1963.
4. Viterbi, Andrew J.: *Principles of Coherent Communication*, McGraw-Hill, New York, 1966.
5. Frazier, J. P., and J. Page: "Acquisition and Tracking Behaviour of Phase-Locked Loops," *IRE Trans. Space Electr. Telem.*, Vol. SET-8, September 1962, pp. 210–227.
6. Sanneman, R. W., and J. R. Rowbotham: "Unlock Characteristic of the Optimum Type II Phase-Locked Loop," *IEEE Trans. Aerosp. Navig. Electron.*, Vol. ANE-11, March 1964, pp. 15–24.
7. *Phase-Locked Loop Data Book*, 2d ed., Motorola Semiconductor Products Inc., Phoenix, AZ, August 1973.
8. Lindsey, William C., and Chak Ming Chie: "A Survey of Digital Phase-Locked Loops," *Proc. IEEE*, Vol. 69, April 1981.
9. Troha, Donald G., and James D. Gallia: "Digital Phase-Locked Loop Design Using SN54/74LS297," *Application Note AN 3216*, Texas Instruments Inc., Dallas, TX.
10. Rohde, Ulrich L.: "Low-Noise Frequency Synthesizers Using Fractional  $N$  Phase-Locked Loops," *r.f. design*, January/February 1981.
11. ———: *Microwave and Wireless Synthesizers, Theory and Design*, John Wiley and Sons, New York, 1997.
12. Hamming, R. W.: *Digital Filters*, 2d ed., Prentice-Hall, Englewood Cliffs, NJ, 1983.
13. Rabiner, L. R., and B. Gold: *Theory and Application of Digital Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1975.
14. Oppenheim, A. V., and R. W. Schafer: *Discrete-Time Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1989.
15. Volgers, R.: "Phase-Locked Loop Circuits: 74HC/HCT4046A & 74HC/HCT7046A," Philips Components, 1989. (Available in the United States from Philips Semiconductors, 811 East Arques Ave., Sunnyvale, CA 94088-3409.)
16. Rosink, W. B.: "All-Digital Phase-Locked Loops Using the 74HC/HCT297," Philips Components, 1989. (Available in the United States from Philips Semiconductors, 811 East Arques Ave., Sunnyvale, CA 94088-3409.)
17. Mullins, Mark: "How to Measure Signal Jitter," *Electronic Products*, July 1992, pp. 41–44.
18. Higgins, Richard J.: *Digital Signal Processing in VLSI*, Prentice-Hall, Englewood Cliffs, NJ, 1990.
19. Kuc, Roman: *Introduction to Digital Signal Processing*, McGraw-Hill, New York, 1988.
20. Sklar, Bernard: *Digital Communications, Fundamentals and Applications*, 2d ed., Prentice-Hall, Upper Saddle River, NJ 07458, 2001.
21. Tzafestas, Spyros G.: *Walsh Functions in Signal and Systems Analysis and Design*, Van Nostrand, New York, 1985.
22. De Bellescize, H.: "La Réception Synchrone," *L' onde Electrique*, Vol. 11, May 1932, pp. 225–240.
23. Best, Roland E.: *Phase-Locked Loops, Theory, Design and Applications*, 1st ed., McGraw-Hill, New York, 1984, Appendix D.
24. Selle, D.: "Theoretische Grundlagen, Dimensionierung und charakteristische Kenngrößen von PLL-Schaltungen (Phasenregelkreisen)," (in German), internal report, available from Prof. Dr. Dieter Selle, Fachhochschule Braunschweig-Wolfenbüttel, Institut für Nachrichtentechnik, D - 3302 Cremlingen (Germany).
25. MATLAB, The MathWorks Inc., 21 Eliot Street, South Natick, MA 01760.

26. Control System Toolbox, a toolbox running under MATLAB (cf. ref. 25).
27. Bently, W. E., and S. G. Varsos: "Squeeze More Data onto Mag Tape by Use of Delay-Modulation Encoding and Decoding," *Electron. Des.*, October 11, 1975.
28. McNamara, John E.: *Technical Aspects of Data Communication*, 1977 & 1978 by Digital Equipment Corporation, Bedford, MA 01730.

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

29. Larimore, W. E.: "Synthesis of Digital Phase-Locked Loops," in *EASCON Rec.*, October 1968, pp. 14–20.
30. Langston, J. Leland: " $\mu$ C Chip Implements High-Speed Modems Digitally," *Electron. Des.*, June 24, 1982.
31. HiJaak 95 Capture, screen Capture Program, part of HiJaak Graphics Suite 95, available from IMSI Technical Support, P.O. Box 3496, Albuquerque, NM 87110-3498, <http://www.imsisoft.com>.
32. 8253 Programmable Interval Timer (Data Sheet), Intel Literature Sales, P.O. Box 58130, Santa Clara, CA 95052-8130.
33. 9513 System Timing Controller (Data Sheet), Advanced Micro Devices Inc., 901 Thompson Place, Sunnyvale, CA 94088.
34. MCS-51 Microprocessor Family, Intel Literature Sales, P.O. Box 58130, Santa Clara, CA 95052-8130.
35. Signal Processing Toolbox, a toolbox running under MATLAB (cf. ref. 25).
36. Brigham, E. O.: *The Fast Fourier Transform*, Prentice-Hall, Englewood Cliffs, NJ, 1974.
37. Tou, Julius T.: *Digital and Sampled-Data Control Systems*, McGraw-Hill, New York, 1959.
38. Borland Delphi 7 Studio, Borland USA, 100 Enterprise Way, Scotts Valley, California 95066-3249.
39. Stofka, Marian: "Digital-Only PLL Exhibits No Overshoot," *EDN*, May 26, 1982.
40. Nyquist, H.: "Certain Topics of Telegraph Transmission Theory", *Trans. Am. Inst. Electr. Eng.*, Vol. 47, Apr. 1928, pp. 617–644.
41. David Grieve, private communication. David Grieve analyzed impulse responses of raised cosine and root raised cosine filters by a MathCad worksheet. For details, contact [david\\_grieve@agilent.com](mailto:david_grieve@agilent.com).
42. Data sheet HSP50210 (Digital Costas Loop), Intersil (formerly Harris Semiconductor), download PDF file from <http://www.intersil.com/design/parametrics/partsearch.asp>.
43. Data sheet HSP50307 (Burst QPSK Modulator), Intersil (formerly Harris Semiconductor), download PDF file from <http://www.intersil.com/design/parametrics/partsearch.asp>.
44. Data sheet HSP50110 (Digital Quadrature Tuner), Intersil (formerly Harris Semiconductor), download PDF file from <http://www.intersil.com/design/parametrics/partsearch.asp>.
45. Application Note AN9661.1, Implementing Polyphase Filtering with the HSP50110 (DQT), HSP50210 (DCL) and the HSP43168 (DFF), January 1999, Intersil (formerly Harris Semiconductor), download PDF file from <http://www.intersil.com/design/parametrics/partsearch.asp>.
46. HSP50110/210EVAL DSP Demodulator Evaluation Board, Intersil (formerly Harris Semiconductor), download PDF file from <http://www.intersil.com/design/parametrics/partsearch.asp>.
47. Grieve, David: Einführung in die Messrichtlinien für DVB-C (Kabel), Fernseh und Kino-Technik 51 (1997), Nr. 7 (in German). For an English translation, contact David Grieve at [david\\_grieve@agilent.com](mailto:david_grieve@agilent.com).
48. Rohde, Ulrich L.: *Microwave and Wireless Synthesizers, Theory and Design*, John Wiley & Sons, New York, 1997.
49. Rohde, Ulrich L., and J. Whitaker: Communication Receivers, DSP, Software Radios, and Design, 3d ed., McGraw-Hill, New York, 2001.
50. EasyPLL, a web application for PLL frequency synthesizer design created by National Semiconductor, <http://www.national.com/appinfo/wireless>.
51. Philips Semiconductor, Data sheet of 74HCT9046A, can be downloaded from <http://philipslogic.com/products/plls/9046>.
52. CMOS phase-locked loops: 74HC(T)4046A/7046A & 74HCT9046A, HCMOS Designer's Guide – advance information, Revised ed.: June 1995, including diskette (3.5") with design program (DOS application). Document order number 9397 750 00078, Philips Semiconductor, 811 East Arques Avenue, Sunnyvale, CA 94088-3409.
53. Pfaff, Dirk: Frequency Synthesis for Wireless Transceivers, dissertation No. 15234, Swiss Federal Institute of Technology, Zurich (Switzerland), 2003.
54. National Semiconductor: LMX2470 2.6 GHz Delta-sigma Fractional-N PLL with 800 MHz Integer-N PLL, Data Sheet, 2003, <http://www.national.com>.
55. Norsworthy, Steven, Schreier R. and Temes G: *Delta-Sigma Data Converters, Theory, Design, and*

- Simulation*, IEEE Press, Piscataway, N.J., 1997.
- 56. Kuo, Tai-Haur, Chen, K. and Chen, J: "Automatic Coefficients Design for Higher-Order Sigma-Delta Modulators," *IEEE Transactions on Circuits and Systems – II: Analog and Digital Signal Processing*, Vol. 46, No. 1, January 1999.
  - 57. Moschytz G. S. and Horn P.: *Active Filter Design Handbook*, John Wiley and Sons, 1981.
  - 58. Shu K., Sanchez-Sinenico E., Silva-Martinez J., and Embabi S.: "A 2.4-GHz Monolithic Fractional-N Frequency Synthesizer with Robust Phase-Switching Prescaler and Loop Capacitance Multiplier," *IEEE Journal of Solid-State Circuits*, Vol. 38, No. 6, June, 2001.

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

59. Data Sheet SAA7320 Stereo CMOS DAC for compact disc digital audio systems (obsolete part), Philips Semiconductor, 811 East Arques Avenue, Sunnyvale, CA 94088-3409.
60. Shu K., Sanchez-Sinencio E., Silva-Martinez J., and Embabi s.: "A 2.4 GHz Monolithic Ractional-N Frequency Synthesizer with Robust Phase-Switching Prescaler and Loop Capacitance Multiplier," *IEEE Journal of Solid-State Circuits*, Vol. 38, No. 6, June 2003.
61. Richard Schreier: Delta-Sigma Toolbox, Toolbox for use with Matlab, can be downloaded from, <http://www.mathworks.com/matlabcentral/fileexchange/loadAuthor.do?objectId=65575&objectType=author>.
62. Ning He, Kuhlmann F. and Buzo A.: "Double loop Sigma-Delta Modulation with DC Input," *IEEE Transactions on Communications*, Vol. 38, No. 4, April 1990.
63. Ulrich L. Rohde and David P. Newkirk: *RF/Microwave Circuit Design for Wireless Applications*, John Wiley & Sons, New York, 2000.

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

# Index

Boldface numbers indicate topics discussed in detail.

$\pi/4$  DQPSK (differential quadrature phase shift keying), [348](#)

AC component, [3](#), [66](#)

AC integrator, [342](#)

ACCU (Accumulator, digital), [164](#)

Acquisition process, [53](#)

ADC (Analog-to-digital converter), [166](#), [274](#), [324](#)

Added White Gaussian Noise (AWGN), [94](#)

Addition theorem:

of the Laplace transform, [413](#)

ADPLL (*see* Phase-locked loop, all-digital)

Algorithm:

filter, [281](#)

SPLL, [330](#), [333](#)

Aliasing, [265](#), [440](#), [445](#)

AM (*see* Amplitude modulation)

Amplitude modulation, [203](#), [341](#)

Amplitude noise, [95](#)

Amplitude response, [358](#), [361](#)

Amplitude shift keying, [343](#)

Analogy (mechanical), [54](#)

Antibacklash circuit, [151](#)

ASK (*see* Amplitude shift keying)

Autocorrelation, [366](#)

Averaging, [264](#)

Backlash (in phase comparators), [151](#)

Bandpass modulation, [341](#)

Bandwidth efficiency, [382](#)

Baseband (transmission), [341](#)

Baud rate, [344](#)

BCD:

counter, [162](#)

counter, fractional, [164](#)

Bellescize, Henri de, [5](#)

BER (*see* Bit error rate)

Bessel function, [413](#)

Binary phase shift keying, [343](#)

Bit cell, [205](#)

Bit error rate (BER), [382](#)

Bit gain, [169](#)

Bode:

diagram, [44](#), [45](#), [97](#), [214](#), [218](#), [261](#)

plot, [261](#)

Borrow, [279](#), [283](#), [293](#)

BPSK (*see* Binary phase shift keying)

Brickwall filter, [358](#), [368](#)

Butterworth filter, [178](#), [194](#)

Capacitor (parasitic), [152](#), [271](#)

Capture range, [62](#)

Carrier signal, [341](#), [343](#), [345](#), [351](#)

Carry, [278](#), [283](#), [293](#)

Cauchy, [435](#)

Causality, [187](#), [448](#)

CB (citizens band), [120](#)

CCO (*see* Current-controlled oscillator)

Center frequency, [3](#), [260](#)

Channel capacity, [344](#), [345](#)

Chebyshev:

filter (inverse), [178](#), [187](#), [192](#), [195](#)

polynomial, [286](#), [456](#)

Check box, [265](#), [266](#)

CMOS (Complementary MOS logic), [124](#), [128](#)

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).

Copyright ©2004 The McGraw-Hill Companies. All rights reserved.

Any use is subject to the Terms of Use as given at the website.

## Code:

bi-phase, [204](#), [207](#)  
 delay modulation, [204](#), [207](#)  
 NRZ, [204](#), [205](#)  
 RZ, [204](#), [205](#)

Communication (coherent), [5](#), [362](#), [366](#), [379](#)

Complex signal, [345](#)

Continuous-phase frequency shift keying, [351](#)

Convergence, [406](#)

## Convolution:

complex, [414](#)  
 integral, [414](#)  
 real, [414](#)

Correlation, [365](#)

Correlation filter, [365](#)

Correlator, [366](#)

Costas loop, [364](#), [373](#)

## Counter:

BCD, [131](#)  
 binary, [131](#)  
 divide-by-N, [9](#), [36](#), [120](#), [162](#)  
 down, [278](#)  
 ID (increment/decrement), [282](#), [294](#)  
 programmable divide-by-N, [120](#)  
 reversible (up/down), [277](#)  
 up, [278](#)

Cross correlation, [367](#)

CSR (*see* cycle slip reduction)

Current-controlled oscillator, [1](#)

Cycle slip reduction, [89](#)

DAC (Digital-analog converter), [164](#), [181](#), [373](#)

Damping factor, [29](#), [42](#), [55](#), [85–88](#), [467](#)

Damping function, [407](#)

DC component, [363](#)

DCO (digital-controlled oscillator), [271](#), [282](#), [324](#)

Delay, [152](#), [328](#), [359](#), [415](#)

Delta function, [58](#), [360](#), [417](#), [433](#)

Delta-Sigma (*see* Sigma-Delta)

Demodulator, [4](#), [362](#), [375](#)

Detector gain, [3](#), [14](#), [463](#)

Dialog box, [257](#), [259](#), [260](#), [261](#), [262](#), [263](#),

Differential equation, [54](#), [397](#)

Differential PSK (phase shift keying), [347](#)

Differentiation, [415](#)

Differentiator, [342](#), [422](#)

Digital video broadcasting, [375](#)

Dither, [175](#), [196](#), [200](#)

Dithering, [175](#), [201](#)

DPLL (*see* Phase-locked loop, digital)

Drift, [271](#)

DSP (Digital signal processor), [321](#)

Duty factor (duty cycle), [296](#), [297](#)

DVB (*see* Digital video broadcasting)

Early-late gate, [371](#)  
Edge detector, [209](#), [332](#)  
Energy per bit ( $E_b$ ), [383](#)  
Energy per symbol ( $E_s$ ), [383](#)  
Error band, [455](#)  
Error checking, [375](#)  
Error function:  
    of filters, [457](#)  
    mathematical, [413](#)  
Error sequence, [168](#), [174](#), [188](#)  
Error transfer function, [43](#), [145](#), [302](#)  
Euler's relations, [443](#)  
Excess bandwidth, [361](#), [377](#)  
EXOR gate, [13](#), [16](#), [64](#), [68](#), [75](#), [84](#), [291](#), [293](#), [298](#), [302](#)  
Extrapolation, [326](#)

Fading, [103](#), [346](#)  
Fastlock technique, [87](#)  
Feedback compensation (weighted), [179](#)  
Feedforward summation, [179](#)  
Filter:  
    active lead-lag, [30](#), [67](#), [74](#), [77](#), [78](#), [79](#), [81](#), [225](#), [235](#), [247](#)  
    active PI, [31](#), [43](#), [74](#), [77](#), [79](#), [227](#), [238](#), [251](#)  
    coefficient, [281](#), [437](#)  
    digital, [281](#), [328](#), [431](#)  
    equiripple, [455](#)  
    FIR, [358](#), [366](#), [379](#), [433](#), [446](#)  
    IIR, [185](#), [433](#)  
    linear-phase, [448](#)  
    low-pass, [3](#), [28](#), [44](#)  
    nonrecursive, [446](#)  
    passive lead-lag, [29](#), [42](#), [64](#), [67](#), [73](#), [77](#), [221](#), [230](#), [242](#)  
    raised cosine, [361](#), [367](#)  
    recursive, [437](#)

Filter (*Cont.*)  
 root raised cosine, [368](#)  
 zero-phase, [359](#), [448](#)

Final value theorem, [48](#), [419](#)

Flipflop:  
 D-, [22](#), [307](#)  
 JK- (edge-triggered), [19](#), [289](#)  
 RS- (edge-triggered), [272](#)

Flicker noise, [140](#), [149](#),  
[327](#), [329](#), [334–336](#)

Flowchart, [327](#), [329](#), [334–336](#)

FM (*see* Frequency modulation)

Fourier:  
 integral, [406](#)  
 spectrum, [406](#)  
 transform, [403](#), [447](#)  
 transform, inverse, [410](#)

Frame synchronization, [357](#)

Frequency:  
 complex, [403](#), [429](#)  
 deviation (*see* Frequency, offset)  
 divider, [9](#), [36](#), [120](#)  
 domain, [301](#)  
 error, [21](#)  
 modulation, [4](#), [66](#), [203](#), [341](#), [467](#)  
 natural, [42](#), [55](#), [467](#)  
 offset, [57](#), [72](#), [389](#)  
 ramp, [49](#)  
 response, [192](#), [361](#), [368](#), [437](#), [448](#), [450](#)

Frequency hopping, [119](#)

Frequency shift keying, [348](#), [377](#), [383](#)

Frequency step, [48](#), [264](#)

Frequency synthesis, [119](#), [159](#)

Frequency synthesizer:  
 fractional- $N$ , [159](#)  
 integer- $N$ , [119](#)

Friction, [54](#)

FSK (*see* Frequency shift keying)

FSK demodulator (decoder), [305](#), [318](#), [377](#)

Gauss filter, [354–355](#)

Gaussian minimum shift keying, [351](#), [383](#)

GMSK (*see* Gaussian minimum shift keying)

Harvard architecture, [322](#)

Help (on simulation), [268](#)

High-gain loop, [43](#)

Hilbert transformer, [274–276](#), [291](#), [454](#)

Hold range, [58](#), [60](#), [62](#), [298](#), [465](#)

IC Master, [385](#)

IIR filter, [184](#), [433](#)

Imbalance (of current), [155](#)

Impedance, [421](#)

Impulse response, [358](#), [361](#), [368](#), [424](#), [431](#), [446](#)

Initial value theorem, [419](#)  
Initialization (of variables), [328](#), [329](#), [334](#), [338](#)  
In-lock detector (*see* lock detector)  
In-phase carrier, [345](#)  
In-phase signal, [345](#)  
Integrate and dump circuit, [369–370](#)  
Integration, [415](#)  
Integrator:  
    general, [104](#)  
    ideal, [397](#)  
Interpolation, [182–183](#)  
Interrupt, [328](#), [333](#)  
Inter-symbol interference, [360](#), [367](#)  
ISI (*see* Inter-symbol interference)

Jitter, [4](#), [95](#), [134](#), [267](#), [295](#), [303](#), [309](#), [315](#), [319](#)

Lagrange:  
    interpolation, [457](#)  
    polynomial, [457](#)  
Laplace:  
    integral, [408](#)  
    transform, [39](#), [47](#), [302](#), [403](#)  
    transform, inverse, [49](#), [410](#)  
Lobe:  
    main, [440](#)  
    side, [440](#), [451](#)  
Lock detector, [104–106](#)  
Lock range, [61](#), [65](#)  
Locked state, [39](#)  
Lock-in process, [53](#), [61](#), [65](#)  
Lock-in time, [62](#)  
Loop filter:  
    digital, [277](#)  
    K-counter, [278](#), [291](#)  
    N-before-M counter, [280](#)  
Low-gain loop, [43](#)  
LPLL (*see* Phase-locked loop, linear)

m-ary FSK (Frequency shift keying), [350](#), [383](#)  
 m-ary PSK (Phase shift keying), [347](#)  
 Magnitude (of frequency response), [43](#), [408](#)  
 MASH converter (multi-stage noise shaping), [197](#)  
 Matched filter, [365](#), [369](#)  
 MATLAB (program), [215](#), [362](#)  
 Maximum likelihood (detector), [367](#)  
 Minimum shift keying, [351](#), [383](#)  
 Mixer, [126](#), [127](#), [131](#)  
 Mobile phone, [381](#)  
 Modulation:  
   AM, [4](#), [203](#), [341](#)  
   FM, [4](#), [66](#), [203](#), [341](#), [467](#)  
   PM, [4](#), [12](#), [203](#), [341](#), [342](#)  
 Motor, [210](#)  
 MPEG-2 standard, [375](#)  
 MSK (*see* Minimum shift keying)  
 Multi-loop synthesizer, [131](#)  
 Multiplier:  
   analog (*see* Multiplier, four-quadrant)  
   digital, [324](#)  
   four-quadrant, [13](#)  
   frequency (up converter), [126](#)

Newton (laws of), [54](#)  
 Noise:  
   bandwidth, [96](#), [99](#), [266](#)  
   Gaussian, [94](#)  
   power, [94](#), [96](#), [136](#), [172](#)  
   power gain, [176](#)  
   shaping, [173](#)  
   in signals, [4](#), [94](#), [134](#), [167](#), [365](#)  
   spectrum, [96](#)  
   suppression, [52](#), [100](#)  
   transfer function, [171](#), [185](#), [197](#)  
 Noise to carrier ratio (NCR), [139](#)  
 Normalization, [335](#)  
 NRZ code, [204](#)  
 Nyquist:  
   frequency, [361](#), [440](#)  
   theorem, [265](#), [440](#)

Offset quadrature phase shift keying, [346](#), [372](#)  
 Optocoupler, [210](#)  
 OQPSK (*see* Offset quadrature phase shift keying)  
 Oversampling, [166](#), [181–183](#), [195](#)

Parabola, [417](#)  
 Parameters (of simulation), [263](#)  
 Parks-McClellan algorithm, [454](#)  
 PD (*see* Phase detector)  
 Pendulum (mathematical), [54](#)  
 PFD (*see* phase-frequency detector)

Phase (of signal), [10](#)  
Phase comparator, [3](#)  
*(see also Phase detector)*  
Phase detector:  
    EXOR, [16](#), [64](#), [68](#), [75](#), [84](#), [155](#), [291](#), [293](#), [296](#)  
    Flipflop counter, [272](#)  
    Hilbert-transform, [274–276](#)  
    JK-Flipflop, [18](#), [64](#), [70](#), [77](#), [84](#), [155](#), [272](#), [296](#), [302](#)  
    Multiplier (four-quadrant multiplier), [13](#), [63](#), [65](#), [71](#), [83](#)  
    Nyquist-rate, [273](#)  
    PFD (phase-frequency detector), [20](#), [64–65](#), [70](#), [79](#), [79](#), [84](#), [151](#), [155](#), [330](#)  
    Zero crossing, [274](#)  
Phase error, [3](#), [14](#), [43](#), [47](#), [40](#), [50](#), [54](#)  
Phase-frequency detector, [20](#), [64–65](#), [70](#), [79](#), [79](#), [84](#), [151](#), [155](#), [330](#)  
Phase jitter, [4](#), [95](#), [134](#), [267](#), [295](#), [303](#), [309](#), [315](#), [319](#)  
Phase margin, [215](#), [219](#)  
Phase modulation, [4](#), [12](#), [203](#), [341](#)  
Phase noise, [95](#), [98](#), [134](#), [187](#)  
Phase perturbation, [138](#)  
Phase response, [215](#)  
Phase shift keying, [343](#)  
Phase step, [47](#), [264](#)  
Phase synchronization, [356](#)  
Phase tracking, [44](#)  
Phase transfer function, [41](#)  
Phase-locked loop:  
    all-digital, [6](#), [271](#)  
    digital, [6](#), [39](#), [330](#)  
    linear, [1](#), [39](#), [324](#)  
    software, [6](#), [321](#)  
Philips, [154](#), [131](#), [385](#)  
Pipeline architecture, [322](#)  
PM (*see* Phase modulation)  
Pole (of transfer function), [29](#), [52](#), [408](#), [435](#)  
Pole-zero plot, [426](#), [426](#)

Power (density) spectrum, [96](#), [137](#), [144](#), [145](#), [148](#)  
 Prefilter, [98](#)  
 Prescaler:  
   2-modulus, [122](#)  
   4-modulus, [124](#)  
   general, [121](#)  
 Prewarping, [442](#)  
 Propagation delay, [207](#)  
 Pseudo corner frequency, [442](#)  
 Pseudo frequency, [442](#)  
 Pseudo-ternary code, [343](#)  
 PSK (*see* Phase shift keying)  
 Pull-in process, [60](#), [389](#)  
 Pull-in range, [60](#), [71](#), [389](#), [465](#)  
 Pull-in time, [62](#), [71](#), [396](#), [401](#)  
 Pull-out range, [61](#), [83](#)  
 Pulse-removing circuit, [164](#)  
 Pushbutton, [181](#), [247](#)

QAM (*see* Quadrature amplitude modulation)  
 QPSK (*see* Quadrature phase shift keying)  
 Quadrature amplitude modulation, [355](#), [375](#)  
 Quadrature carrier, [345](#)  
 Quadrature FSK, [350](#)  
 Quadrature phase shift keying, [345](#), [372](#)  
 Quadrature signal, [274](#)  
 Quantization:  
   error, [167](#)  
   step, [193](#)  
 Quantizer, [166](#)  
 Quartz crystal, [36](#), [117](#), [120](#)  
 Quaternary phase shift keying, [345](#)

Radio button, [259](#)  
 Raised cosine filter, [361](#), [367](#)  
 Ramp function, [12](#), [48](#)  
 Receiver, [367](#), [373](#)  
 Reciprocal mixing, [135](#)  
 Redundancy, [375](#)  
 Reed-Solomon code, [377](#)  
 Remez algorithm, [450](#), [454](#)  
 Residue, [427](#), [435](#)  
 Residue theorem, [435](#)  
 Ripple:  
   frequency, [155](#)  
   reduction, [303](#)  
 Ripple (of signals), [150](#), [294](#), [297](#), [303](#), [453](#), [455](#)  
 Root raised cosine filter, [368](#)  
 RS latch, [32](#)  
 RZ code, [205](#)

Sampling:  
   frequency, [265](#), [332](#), [432](#), [445](#)  
   function, [435](#), [437](#)

interval, [281](#), [286](#)  
theorem, [265](#), [432](#)  
Saturation, [18](#), [20](#), [23](#), [260](#)  
Schmitt trigger, [106](#), [210](#)  
Scrollbar, [266](#)  
*s*-domain (*see s*-plane)  
Servo amplifier, [45](#), [210](#)  
Settling time, [68](#)  
Shannon theorem, [265](#), [441](#)  
Sideband suppression, [157](#)  
Sigma-Delta:  
    ADC, [166](#)  
    DAC, [181](#)  
    modulator, [183](#)  
Signal:  
    power, [95](#)  
    transfer function, [171](#), [186](#)  
Signal-to-noise ratio, [98](#), [266](#)  
Simulation:  
    of ADPLL, [311](#)  
    of DPLL, [257](#)  
    of LPPLL, [257](#)  
sinc function, [358](#), [451](#)  
single-loop synthesizer, [131](#)  
SNR (*see* Signal-to-noise ratio)  
*s*-plane (*s*-domain), [408](#), [435](#)  
SPLL (*see* Phase-locked loop, software)  
Spread-spectrum technique, [119](#)  
Spur (*see* spurious sidebands)  
Spurious sidebands, [150](#)  
Squaring loop, [363](#)  
Stability:  
    dynamic, [59](#)  
    of feedback systems, [211](#), [217](#)  
    static, [61](#)  
Staircase signal, [164](#), [369](#)

State diagram (of logic devices), [22](#)  
 State transition, [22](#)  
 Steady-state error, [50](#)  
 Structogram, [327](#), [329](#), [334–336](#)  
 Subharmonic (signals in phase comparators), [152](#)  
 Sweep technique, [104](#)  
 Switched-filter technique, [106](#)  
 Symbol:  
   period, [359](#), [371](#)  
   rate, [344](#), [369](#), [375](#)  
   synchronization, [357](#), [369–371](#)

Tachometer, [210](#)  
 Tangent function:  
   hyperbolic, [442](#)  
   trigonometric, [442](#)  
 Tangent transform, [442](#)  
 Tap, [446](#)  
 Taylor series, [393](#)  
 Temperature drift, [271](#)  
 Thermal noise, [138](#)  
 Time domain, [403](#)  
 Timer, [339](#)  
 Timer/counter, [333](#)  
 Torque, [54–55](#)  
 Tracking, [44](#)  
 Transfer function, [39](#), [41](#), [46](#), [302](#), [426](#), [431](#)  
 Transform:  
   Fourier, [403](#)  
   Laplace, [406](#)  
    $z$ -bilinear, [441](#)  
    $z$ -impulse invariant, [434](#)  
 Transient response, [47](#), [424](#)  
 Transition, [453](#)  
 Transition frequency, [219](#)  
 Transmission:  
   baseband, [203](#), [341](#)  
   carrier-based, [203](#), [341](#)  
 Transmitter, [357](#), [372](#)

Unit step function, [12](#), [410](#)  
 Unlocked state, [53](#)

VCO (*see* Voltage-controlled oscillator)  
 VCO gain, [3](#), [33](#), [462](#)  
 Voltage-controlled oscillator, [1](#), [32–37](#)  
 Voltage-to-current converter, [33](#)

Web (World Wide Web), [385](#)  
 Window:  
   Bartlett, [453](#)  
   Blackman, [453](#)  
   flattop, [453](#)

function, [448](#)  
Hamming, [453](#)  
Hanning, [453–454](#)  
Kaiser, [453](#)  
rectangular, [450](#)  
triangular, [453](#)

*z*-domain (*see z*-plane)  
Zero (of transfer function), [29](#), [408](#)  
Zero-crossing technique, [274](#)  
Zooming (in PLL simulation), [266](#)  
*z*-operator, [281](#)  
*z*-plane, [433](#)  
*z*-transform:  
    bilinear, [441](#)  
    general, [433](#)  
    impulse-invariant, [434](#)

---

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.

## CD-ROM WARRANTY

This software is protected by both United States copyright law and international copyright treaty provision. You must treat this software just like a book. By saying "just like a book," McGraw-Hill means, for example, that this software may be used by any number of people and may be freely moved from one computer location to another, so long as there is no possibility of its being used at one location or on one computer while it also is being used at another. Just as a book cannot be read by two different people in two different places at the same time, neither can the software be used by two different people in two different places at the same time (unless, of course, McGraw-Hill's copyright is being violated).

### LIMITED WARRANTY

Customers who have problems installing or running a McGraw-Hill CD should consult our online technical support site at <http://books.mcgraw-hill.com/techsupport>. McGraw-Hill takes great care to provide you with topquality software, thoroughly checked to prevent virus infections. McGraw-Hill warrants the physical CD-ROM contained herein to be free of defects in materials and workmanship for a period of sixty days from the purchase date. If McGraw-Hill receives written notification within the warranty period of defects in materials or workmanship, and such notification is determined by McGraw-Hill to be correct, McGraw-Hill will replace the defective CD-ROM. Send requests to:

McGraw-Hill  
Customer Services  
P.O. Box 545  
Blacklick, OH 43004-0545

The entire and exclusive liability and remedy for breach of this Limited Warranty shall be limited to replacement of a defective CD-ROM and shall not include or extend to any claim for or right to cover any other damages, including, but not limited to, loss of profit, data, or use of the software, or special, incidental, or consequential damages or other similar claims, even if McGraw-Hill has been specifically advised of the possibility of such damages. In no event will McGraw-Hill's liability for any damages to you or any other person ever exceed the lower of suggested list price or actual price paid for the license to use the software, regardless of any form of the claim.

**McGRAW-HILL SPECIFICALLY DISCLAIMS ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, ANY IMPLIED WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.**

Specifically, McGraw-Hill makes no representation or warranty that the software is fit for any particular purpose and any implied warranty of merchantability is limited to the sixty-day duration of the Limited Warranty covering the physical CD-ROM only (and not the software) and is otherwise expressly and specifically disclaimed.

This limited warranty gives you specific legal rights; you may have others which may vary

from state to state. Some states do not allow the exclusion of incidental or consequential damages, or the limitation on how long an implied warranty lasts, so some of the above may not apply to you.

---

Printed from Digital Engineering Library @ McGraw-Hill ([www.Digitalengineeringlibrary.com](http://www.Digitalengineeringlibrary.com)).  
Copyright ©2004 The McGraw-Hill Companies. All rights reserved.  
Any use is subject to the Terms of Use as given at the website.