McGRAW-HILL
PROFESSIONAL ENGINEERING

- CD-ROM of new software for the design of entire PLL systems up to order 5
- Step-by-step procedures for design of linear and digital PLL circuits
- Simple method for designing higher-order PLL systems
- Ready-to-use design examples for digital PLL frequency synthesizers
- New directory of commercially available PLL IC's

FIFTH EDITION

Phase-Locked Loops

DESIGN, SIMULATION, AND APPLICATIONS

ROLAND E. BEST

Phase-Locked Loops

Design, Simulation, and Applications

Roland E. Best
Best Engineering
Oberwil, Switzerland

Fifth Edition

McGraw-Hill
New York Chicago San Francisco Lisbon
London Madrid Mexico City Milan New Delhi
San Juan Seoul Singapore Sydney Toronto

The McGraw-Hill Companies

Library of Congress Cataloging-in-Publication Data

Best, Roland E.
Phase-locked loops / Roland E. Best—5th ed.
p. cm.
Includes bibliographical references and index.
ISBN 0-07-141201-8
1. Phase-locked loops. I. Title.

TK7872.P38B475 2003
621.3815'364—dc21 2003046445

Copyright © 2003 by The McGraw-Hill Companies, Inc. All rights reserved. Printed in the United States of America. Except as permitted under the United States Copyright Act of 1976, no part of this publication may be reproduced or distributed in any form or by any means, or stored in a data base or retrieval system, without the prior written permission of the publisher.

1 2 3 4 5 6 7 8 9 0 DOC/DOC 0 9 8 7 6 5 4 3

P/N 141202-6
PART OF
ISBN 0-07-141201-8

The sponsoring editor for this book was Steve Chapman, the editing supervisor was Caroline Levine, and the production supervisor was Sherri Souffrance. It was set in New Century Schoolbook by SNP Best-set Typesetter Ltd., Hong Kong.

Printed and bound by RR Donnelley.

This book is printed on recycled, acid-free paper containing a minimum of 50 percent recycled de-inked fiber.

McGraw-Hill books are available at special quantity discounts to use as premiums and sales promotions, or for use in corporate training programs. For more information, please write to the Director of Special Sales, McGraw-Hill Professional, Two Penn Plaza, New York, NY 10121-2298. Or contact your local bookstore.

Information contained in this work has been obtained by The McGraw-Hill Companies, Inc. ("McGraw-Hill") from sources believed to be reliable. However, neither McGraw-Hill nor its authors guarantee the accuracy or completeness of any information published herein, and neither McGraw-Hill nor its authors shall be responsible for any errors, omissions, or damages arising out of use of this information. This work is published with the understanding that McGraw-Hill and its authors are supplying information but are not attempting to render engineering or other professional services. If such services are required, the assistance of an appropriate professional should be sought.

Contents

Preface to the Fifth Edition

Historically the PLL has been a linear circuit. While the first PLLs were realized with discrete components, they became available as ICs in about 1965. The first of these were linear devices (LPLLs), built in semiconductor technologies similar to the operational amplifiers of that era. A few years later (about 1970) the first "digital" PLLs (DPLLs) became available, but when looking at their schematics, we recognize that only the phase detector was built from logic circuits, while the remaining parts (voltage-controlled oscillator (VCO), loop filter) stayed analog; hence these PLLs must be considered hybrid systems. In this new edition of the book we combined the two categories LPLL and DPLL into one single class named "mixed-signal PLL." Doing so greatly simplifies the analysis, because both classes now can be treated by a unified theory.

Chapter 1 is a short introduction into the PLL domain, and Chap. 2 deals with theory, design, and applications of mixed-signal PLLs. The discussion includes different types of phase detectors (linear and digital), phase-frequency detectors with charge-pump outputs, loop filters (active and passive), and VCOs. Typical mixed-signal PLL applications are given, such as circuits for retiming and clock recovery, motor speed control, and the like.

Because frequency synthesis is one of the most important applications of DPLLs, a separate chapter (3) is devoted to digital PLL frequency synthesizers. Because phase jitter and spurious sidebands are the most disturbing phenomena with frequency synthesizers, different methods are presented to overcome those problems, e.g., antibacklash circuits and higher-order loop filters. Moreover, the analysis covers both integer-N and fractional-N synthesizers and demonstrates that the latter can acquire "lock" much faster, a feature that is used with great benefit in frequency-hopping (spread-spectrum) applications. Spread-spectrum techniques will become increasingly important in the newest generations of mobile phones. It is demonstrated furthermore that simple synthesizers can be realized as single loops, but that multiloop configurations become mandatory in high-performance systems.

Because higher-order systems (filters) are a must in many synthesizer applications, Chap. 4 deals with the design of such systems, i.e., of PLLs up to fifth

order. In designing higher-order loops, the placement of poles and zeros can be a difficult task. The design is greatly facilitated by a novel method developed by the author, based on simple Bode diagrams. To ease system realization even further, a program developed by the author is included on a CD-ROM, which automatically designs and analyses PLLs up to fifth order; this topic is discussed in Chap. 5, which also includes many design examples. Having synthesized a PLL circuit, the program can be used to simulate dynamic performance of the system, e.g., lock-in and lock-out processes. To investigate PLL performance in the presence of noise—which is the normal scenario in practice—the user can superimpose narrowband or broadband noise of any desired level. Finally, the program displays Bode diagrams of the synthesized PLL and loop filter schematics, including the component values.

Chapter 6 treats theory, design and applications of the "all-digital" PLL (ADPLL), a PLL category introduced later than the preceding ones. In contrast to LPLLs and DPLLs, which are continuous-time systems, the ADPLL is a discrete-time device and therefore exhibits relatively large ripple (phase jitter). The applications of the ADPLL are therefore restricted to those cases where ripple can be tolerated, e.g., frequency-shift keying (FSK) decoders and the like. Chapter 7 describes computer-aided design and simulation of ADPLLs, using the already mentioned computer program.

Because the speed of microcontrollers and digital signal processors increased dramatically in recent years, many PLL applications can now be implemented by software. Chapter 8 discusses the hardware/software tradeoff in the domain of the PLL and describes a number of software algorithms that can be used to implement software PLLs (SPLLs).

In Chap. 9 a review of PLL applications in the field of communications is given. It includes the most important digital modulation schemes such as BSK, QPSK, FSK, and QAM and describes a number of special PLL circuits used for carrier and symbol synchronization (e.g., Costas loop, early-late gate, integrate-and-dump circuit) and measures to prevent intersymbol interference (ISI), e.g., root-raised-cosine filters. Among other topics, this chapter also explains how to increase the symbol rate of digital communications without increasing system bandwidth.

Chapter 10 contains a list of PLL ICs currently available from American, European, and Japanese manufacturers, including short descriptions of the circuits. The list includes full PLL systems on a chip, parts of PLLs such as phase detectors and VCOs, and complex systems based on PLL-like frequency synthesizers or radio/TV chips. It also includes many single- and dual-modulus prescalers.

Finally, Chap. 11 demonstrates how the parameters of PLLs can be measured with conventional lab instruments such as oscilloscopes and signal generators.

Three appendixes provide additional information on selected topics. In App. A the analysis of the acquisition process is developed for the mathematically interested reader. Appendix B is a primer on the Laplace transform, which is

frequently used in this book, and App. C is an overview on digital filtering, which has become an increasingly important issue in the design of high-complexity PLL systems.

Roland Best

Chapter

1

Introduction to PLLs

1.1 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 will keep heads at the top of the screen and feet at the bottom. In color television, another PLL makes sure that green remains green and red remains red (even if the politicians claim that 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 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. 1.2, there exist other types of PLLs, e.g., digital PLLs (DPLLs), all-digital PLLs (ADPLLs), and software PLLs (SPLLs). Its block diagram is shown in Fig. 1.1*a*. The PLL consists of three basic functional blocks:

1. A voltage-controlled oscillator (VCO)
2. A phase detector (PD)
3. A loop filter (LF)

In this simple example, there is no down-scaler between output of VCO [$u_2(t)$] and lower input of the phase detector [ω_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

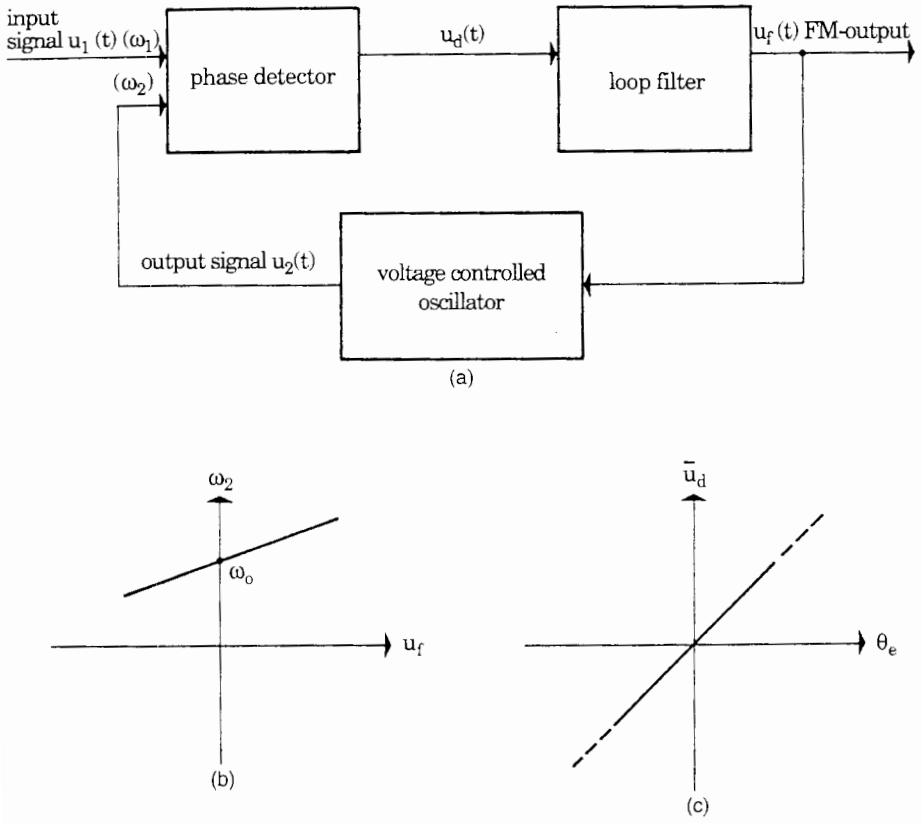


Figure 1.1 (*a*) Block diagram of the PLL. (*b*) Transfer function of the VCO. (u_f = control voltage; ω_2 = angular frequency of the output signal.) (*c*) Transfer function of the PD. ($\overline{u_d}$ = average value of the phase-detector output signal; θ_e = phase error.)

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 ω_1 of the reference signal
- The output signal $u_2(t)$ of the VCO
- The angular frequency ω_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 θ_e, defined as the phase difference between signals $u_1(t)$ and $u_2(t)$

Let us now look at the operation of the three functional blocks in Fig. 1.1*a*. The VCO oscillates at an angular frequency ω_2, which is determined by the output signal u_f of the loop filter. The angular frequency ω_2 is given by

$$\omega_2(t) = \omega_0 + K_0 u_f(t) \tag{1.1}$$

where ω_0 is the center (angular) frequency of the VCO and K_0 is the VCO gain in rad $\cdot$ s^{-1} $\cdot$ V^{-1}.

Equation (1.1) is plotted in Fig. 1.1*b*. Because the 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° must always be specified as a value of π.

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

$$u_d(t) = K_d \theta_e \tag{1.2}$$

Here K_d represents the gain of the PD. The physical unit of K_d is volts per radian. Figure 1.1*c* 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 that the angular frequency of the input signal $u_1(t)$ is equal to the center frequency ω_0. The VCO then operates at its center frequency ω_0. As we see, the phase error θ_e is zero. If θ_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 θ_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 that is greater than its center frequency ω_0 by an amount $\Delta\omega$. This will force the signal $u_f(t)$ to settle at a final 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

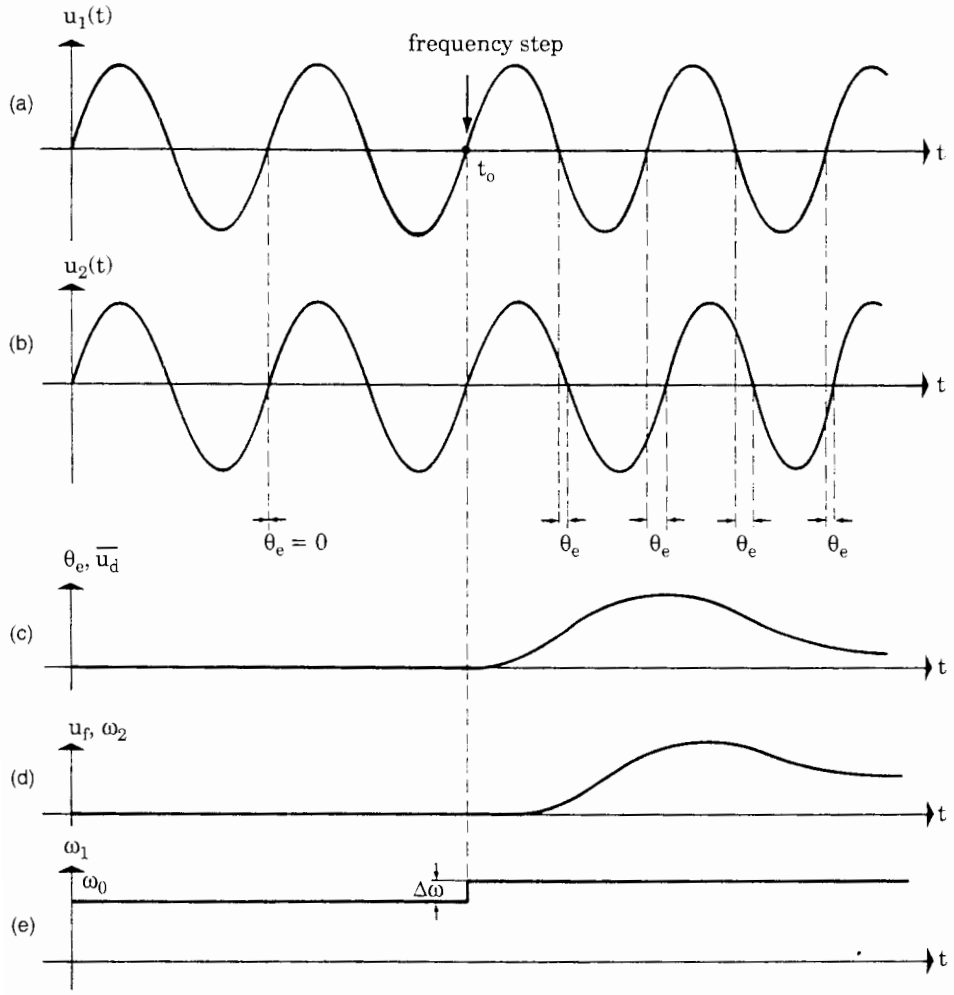


Figure 1.2 Transient response of a PLL onto a step variation of the reference frequency. (*a*) Reference signal $u_1(t)$. (*b*) Output signal $u_2(t)$ of the VCO. (*c*) Signals $\overline{u_d}(t)$ and $\theta_e(t)$ as a function of time. (*d*) Angular frequency ω_2 of the VCO as a function of time. (*e*) Angular frequency ω_1 of the reference signal $u_1(t)$.

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 have to 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 that there are some fundamentally different types of PLLs. Therefore, we first identify these various types.

1.2 Classification of PLL Types

The very first phase-locked loops (PLLs) were implemented as early as 1932 by de Bellescize[22]; this French engineer is considered the inventor of "coherent communication." The PLL found broader industrial applications only when it became available as an integrated circuit. The first PLL ICs appeared around 1965 and were 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 following years 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, e.g., 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 and hence doesn't contain any passive components like resistors and capacitors.

Analogous 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 that covers all kinds of PLLs. The performance of LPLLs and DPLLs is similar, however; hence we can develop a theory that is valid for both categories. We will deal with LPLLs and DPLLs in Chap. 2, "Mixed-Signal PLLs." 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 differently from mixed-signal PLLs and hence is treated in a separate chapter (Chap. 6).

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. 8.

Chapter

2

Mixed-Signal PLLs

2.1 Block Diagram of the Mixed-Signal PLL

As mentioned in Sec. 1.2, 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, we 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 *voltage-controlled oscillator* (VCO). When the PLL is used as a frequency synthesizer, another block is added: a *divide-by-N counter*. Assuming that the counter divides by a factor N, the frequency of the VCO output signal is forced then 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 Chap. 3.

When a down-scaler is inserted, the term *center frequency* becomes ambiguous: the center (radian) frequency ω_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 ω_0 to denote the center frequency at the output of the VCO, and the symbol ω_0' to denote the center radian frequency at the input of the PLL. Obviously, ω_0 and ω_0' are related by $\omega_0' = \omega_0/N$. As seen from Fig. 2.1, the quantities related to the output signal of the down-scaler are characterized by a prime ($'$ symbol), e.g., u_2', ω_2'. When the VCO does not operate at its center frequency ($u_f \neq 0$), its output radian frequency is denoted ω_2. For the down-scaled frequency, the symbol ω_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 second-order PLLs. Higher-order loops come into play when suppression of

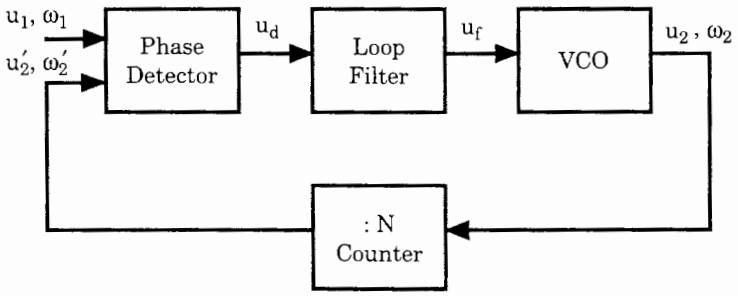


Figure 2.1 Block diagram of the mixed-signal PLL. The symbols defined here are used throughout this chapter. The divide-by-N counter is optional.

spurious sidebands (also called "spurs") becomes an issue. The designer of the PLL then has to provide higher-order loop filters,[10] i.e., 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. 4) 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 that acts on the VCO in such a way that the frequency of the signal u_2' is identical to 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, for example, voltage or current signals, 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. 2.3.

2.2 A Note on Phase Signals

Dynamic analysis of a control system 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

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

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 phase signals, however, which are less familiar to many electronic engineers.

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:

$$u_1(t) = U_{10}\sin[\omega_1 t + \theta_1(t)]$$
$$u_2'(t) = U_{20}\sin[\omega_2' t + \theta_2'(t)] \tag{2.2}$$

The information carried by these signals is neither the amplitude (U_{10} or U_{20}, respectively) nor the frequency (ω_1 or ω_2', respectively) but the phases $\theta_1(t)$ and $\theta_2'(t)$. (*Note*: because we used the symbol ω_2' for the radian frequency at the output of the down-scaler (Fig. 2.1), we use the symbol θ_2' for the phase of signal u_2' and not θ_2; the latter is used to specify the phase of the VCO output signal u_2.)

Let us consider now some simple phase signals; Fig. 2.2 lists a number of phase signals $\theta_1(t)$ that are frequently used to excite a PLL. Figure 2.2*a* shows the simplest case: the phase $\theta_1(t)$ performs a step change at time $t = 0$, hence is given by

$$\theta_1(t) = \Delta\Phi \cdot u(t) \tag{2.3}$$

where $u(t)$ is the unit step function. This case is an example of phase modulation. Let us consider next an example of frequency modulation (Fig. 2.2*b*). Assume the angular frequency of the reference signal is ω_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

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

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

$$\theta_1(t) = \Delta\omega \cdot t \tag{2.5}$$

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

As a last example, consider a reference signal whose angular frequency is ω_0' for $t < 0$ and increases linearly with time for $t \geq 0$ (Fig. 2.2*c*). For $t \geq 0$ its angular frequency is therefore

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

where $\Delta\dot{\omega}$ 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:

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

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

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

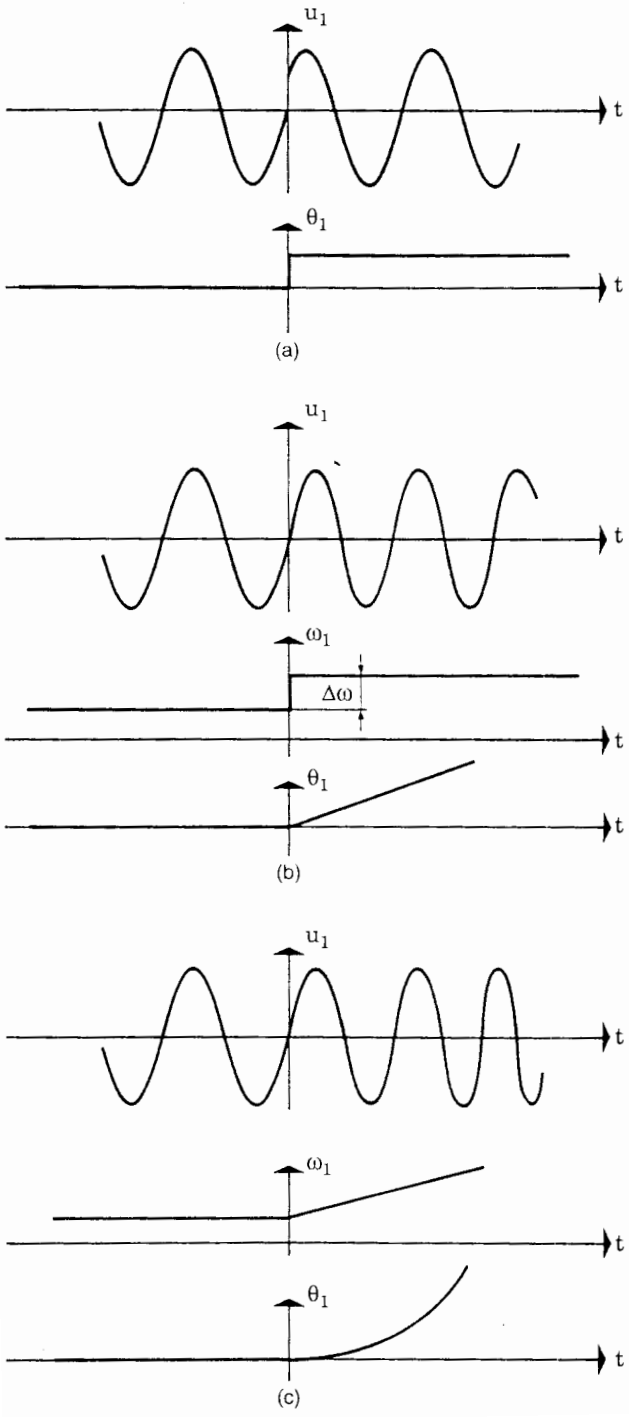


Figure 2.2 Some typical exciting functions as applied to the reference input of a PLL. (*a*) Phase step applied at $t = 0$; $\theta_1(t) = \Delta\Phi \cdot u(t)$. (*b*) Frequency step $\Delta\omega$ applied at $t = 0$; $\theta_1(t) = \Delta\omega \cdot t$. (*c*) Frequency ramp starting at $t = 0$; $\theta_1(t)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

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

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

i.e., it is a quadratic function of time.

2.3 Building Blocks of Mixed-Signal PLLs

The building blocks of a PLL have been defined in Fig. 2.1. In the following sections we are going to discuss the properties of various phase detectors, loop filters, controlled oscillators, and down-scalers.

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

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

where U_{10} is the amplitude of the signal, ω_1 is its radian frequency, and θ_1 its phase. The second input signal u_2' (Fig. 2.1) is usually a symmetrical square wave signal (sometimes also called *Walsh* function) [21] and is given by

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

where rect stands for rectangular (square wave) and U_{20} is the amplitude, ω_2' the radian frequency, and θ_2' the phase. These signals are shown in Fig. 2.3. The dashed curve in Fig. 2.3*a* is a sine wave having a phase of $\theta_1 = 0$; the solid line has a nonzero phase θ_1. For simplicity we assume here that the phase is constant over time. The dashed curve in Fig. 2.3*b* 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

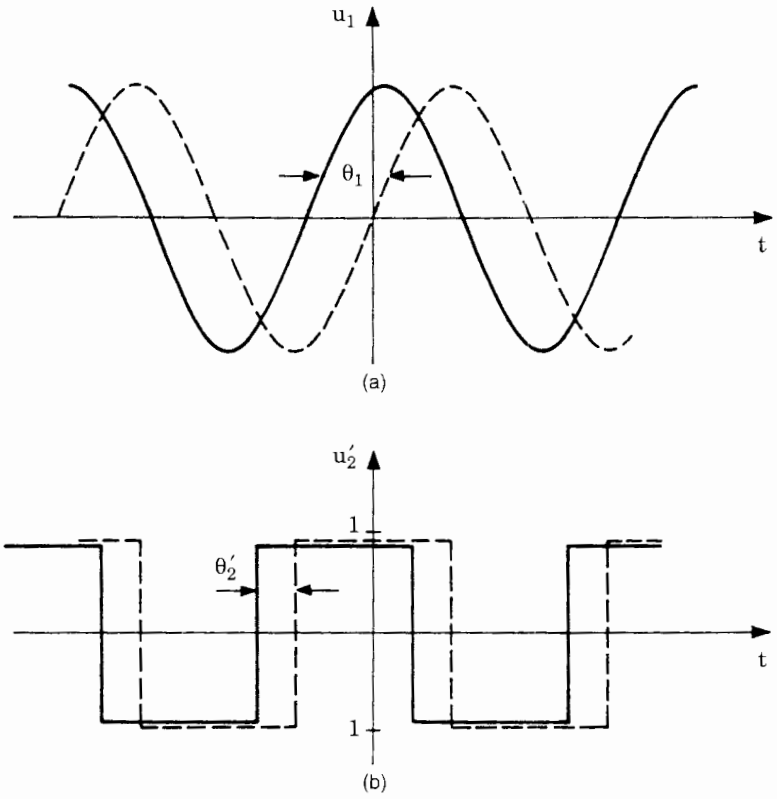


Figure 2.3 Input signals of the multiplier phase detector. (*a*) Signal $u_1(t)$ is a sine wave. Dashed line: phase $\theta_1 = 0$; solid line: phase $\theta_1 > 0$. (*b*) Signal $u_2'(t)$ is a symmetrical square wave signal. Dashed line: $\theta_2' = 0$; solid line: $\theta_2' > 0$.

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

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

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

When the PLL is locked, the frequencies ω_1 and ω_2' are identical, and $u_d(t)$ becomes

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

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

$$u_d(t) \approx K_d \sin(\theta_e) \tag{2.14}$$

where K_d is called detector gain. When the phase error is small, the sine function can be replaced by its argument, and we have

$$u_d(t) \approx K_d\theta_e \tag{2.15}$$

This equation represents the linearized model of the phase detector.

The dimension of K_d is radians per volt (rad/V). As shown above, 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 ω_1 and ω_2' are different. The output signal of the multiplier then can be written as

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

The higher harmonics are almost entirely suppresed by the loop filter; hence there remains one ac term whose frequency equals the difference $\omega_1 - \omega_2'$.

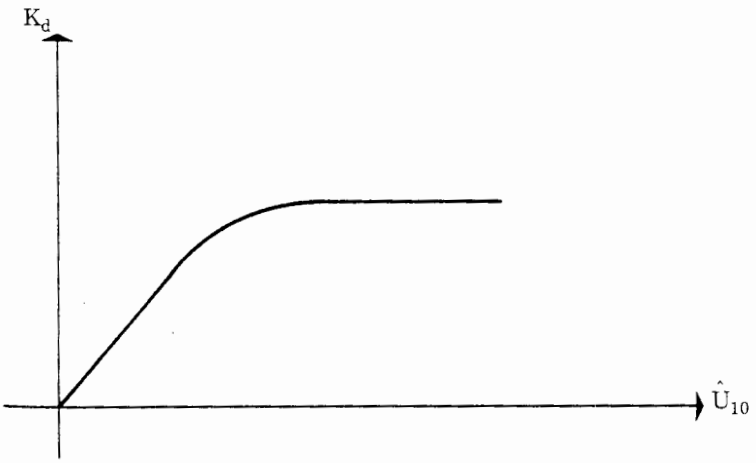


Figure 2.4 Phase-detector gain K_d as a function of the amplitude U_{10} of the reference signal.

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 permanently stay hung up at its center frequency ω_0, with a superimposed frequency modulation. As we will see in "Pull-out Range $\Delta\omega_{PO}$" in Sec. 2.6.2, however, the ac signal $u_d(t)$ is an asymmetric "sine wave"; i.e., 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, let us note that this pull-in process is quite slow. It will be demonstrated later in this section 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 detector. 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, i.e., 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 for different phase errors θ_e. At zero phase error the signals u_1 and u_2' are out of phase by exactly 90°, as shown in Fig. 2.6*a*. 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. 2.6*a*. The average value (denoted $\overline{u_d}$) is the arithmetic mean of the two logical levels; if the EXOR is powered from an asymmetrical 5-V power supply, $\overline{u_d}$ will be approximately 2.5 V. This voltage level is considered the quiescent point of the EXOR and will be denoted as $\overline{u_d} = 0$ from now on. When the output signal u_2' lags the reference signal u_1, the phase error θ_e becomes positive by definition; this case is shown in Fig. 2.6*b*.

Now the duty cycle of u_d becomes larger than 50 percent; i.e., 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

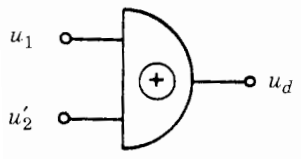


Figure 2.5 EXOR phase detector.

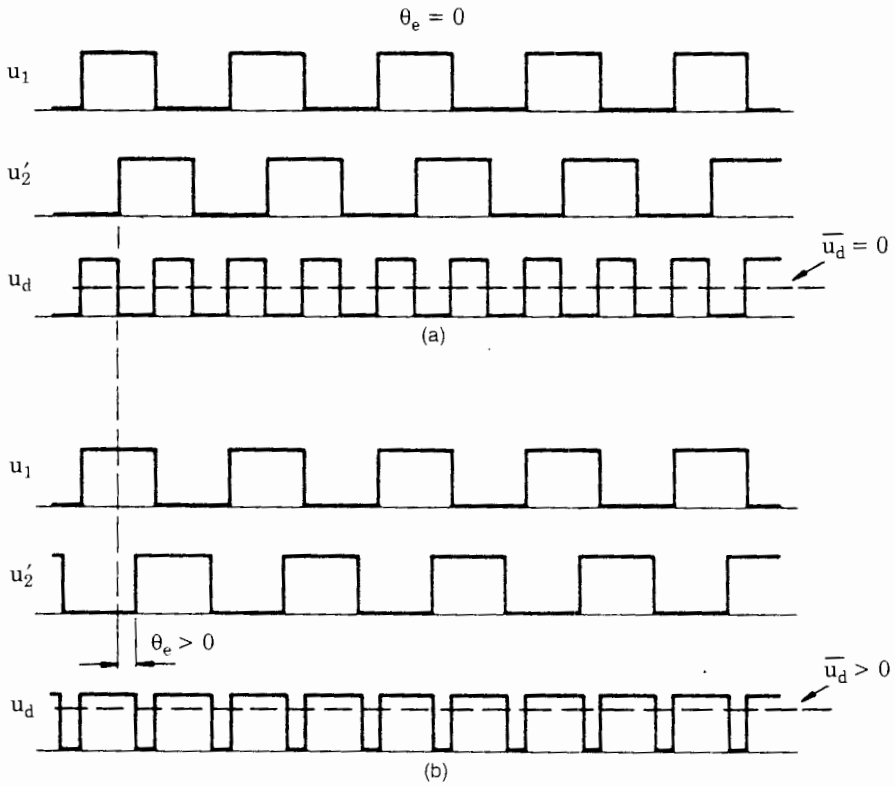


Figure 2.6 Waveforms of the signals for the EXOR phase detector. (*a*) Waveforms at zero phase error ($\theta_e = 0$). (*b*) Waveforms at positive phase error ($\theta_e > 0$).

$\theta_e = 90°$ and its minimum value for $\theta_e = -90°$. If we plot the mean of u_d versus phase error θ_e, we get the characteristic shown in Fig. 2.7*a*. 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 $-90° < \theta_e < 90°$, $\overline{u_d}$ is exactly proportional to θ_e and can be written as

$$\overline{u_d} = K_d \theta_e \tag{2.17}$$

In 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

$$K_d = \frac{U_B}{\pi} \tag{2.18}$$

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

$$K_d = \frac{U_{sat+} - U_{sat-}}{\pi} \tag{2.19}$$

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. 2.7*b*. This reduces the loop gain of the PLL and results in smaller lock range, pull-out range, etc.

It is important to look also 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 ω_1 and ω_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 will be filtered out by the loop filter. The EXOR therefore performs very much like the multiplier phase detector; i.e., the pull-in process becomes slow, and acqui-

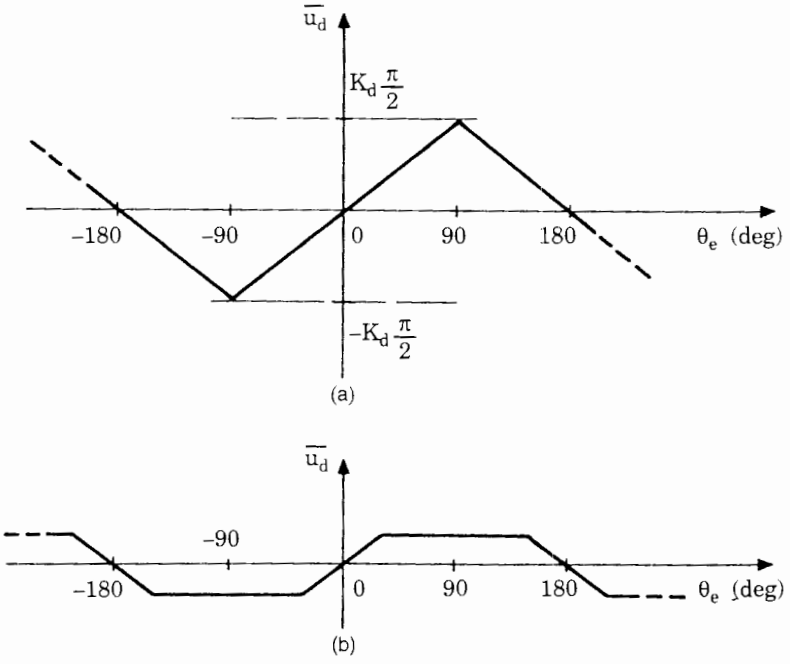


Figure 2.7 Plot of averaged phase detector output signal $\overline{u_d}$ versus phase error θ_e. (*a*) Normal case: waveforms u_1 and u_2' in Fig. 2.6 are symmetrical square waves. (*b*) Waveforms u_1 and u_2' are asymmetrical. The characteristic of the phase detector is clipped.

sition is realized only when the difference ω_1 and ω_2' is less than the pull-in frequency $\Delta\omega_P$. This will be discussed in more detail in "Pull-in Range $\Delta\omega_P$ and Pull-in Time T_P" in Sec. 2.6.2.

Type 3: JK-flipflop phase detector. The JK-flipflop phase detector is shown in Fig. 2.8. 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 ($Q = 1$), a positive edge at the K input into its "low" state ($Q = 0$). Figure 2.9*a* 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

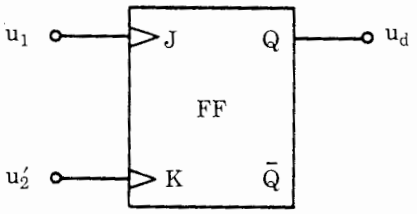


Figure 2.8 JK-flipflop phase detector.

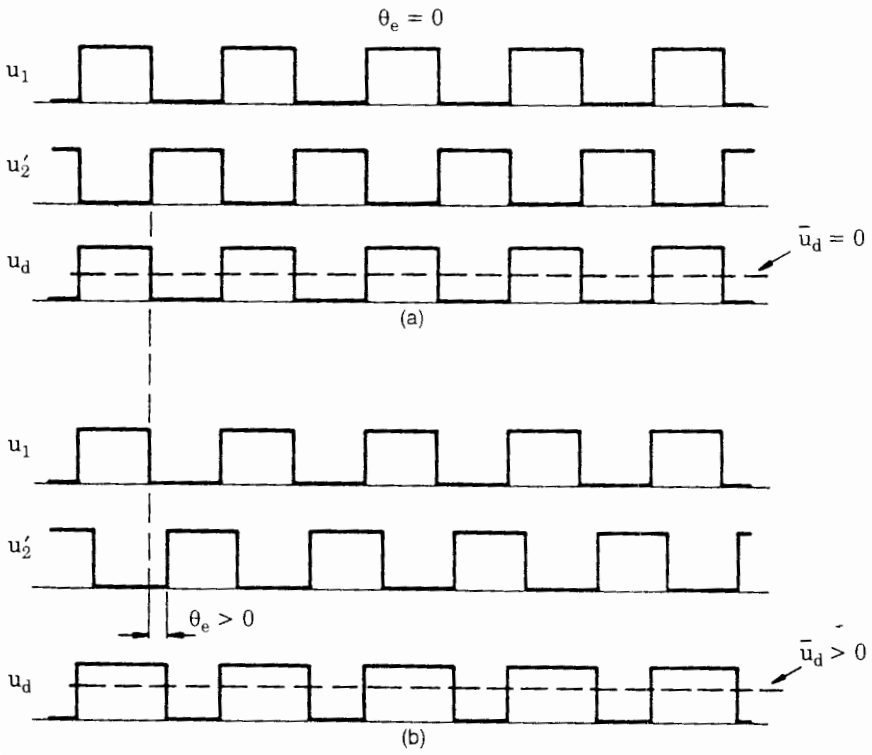


Figure 2.9 Waveforms of the signals for the JK-flipflop phase detector. (*a*) Waveforms at zero phase error. (*b*) Waveforms at positive phase error.

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 $\overline{u_d}$ being zero. If the phase error becomes positive (Fig. 2.9*b*), the duty cycle of the u_d signal becomes greater than 50 percent; i.e., $\overline{u_d}$ becomes positive. Clearly, $\overline{u_d}$ becomes maximum when the phase error reaches 180° and minimum when the phase error is −180°. If the mean value of u_d is plotted versus phase error θ_e, the sawtooth characteristic of Fig. 2.10 is obtained.

Within a phase error range of $-\pi < \theta_e < \pi$ the average signal $\overline{u_d}$ is proportional to θ_e and is given by

$$\overline{u_d} = K_d \theta_e \tag{2.20}$$

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

$$K_d = \frac{U_B}{2\pi} \tag{2.21}$$

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

$$K_d = \frac{U_{sat+} - U_{sat-}}{2\pi} \tag{2.22}$$

In contrast to that for 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.

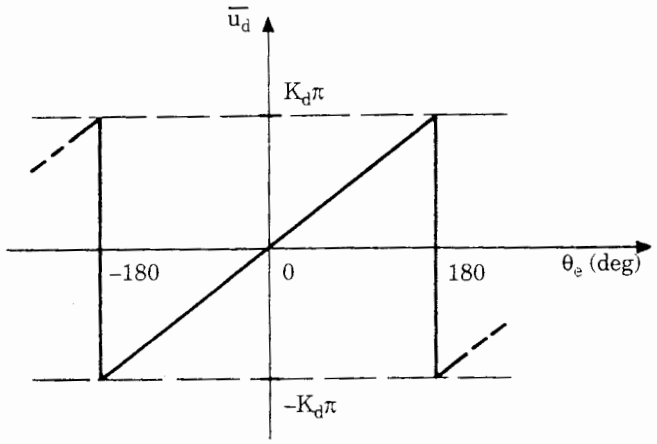


Figure 2.10 Plot of averaged phase detector output signal $\overline{u_d}$ versus phase error θ_e. In contrast to the EXOR, $\overline{u_d}$ does not depend on the duty cycle of the signals.

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 ω_1 and ω_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 "Pull-in Range $\Delta\omega_P$ and Pull-in Time T_P" in Sec. 2.6.2.

Type 4: Phase-frequency detector (PFD). 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 θ_e but also on frequency error $\Delta\omega = \omega_1 - \omega_2'$, when the PLL has not yet acquired lock. The 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

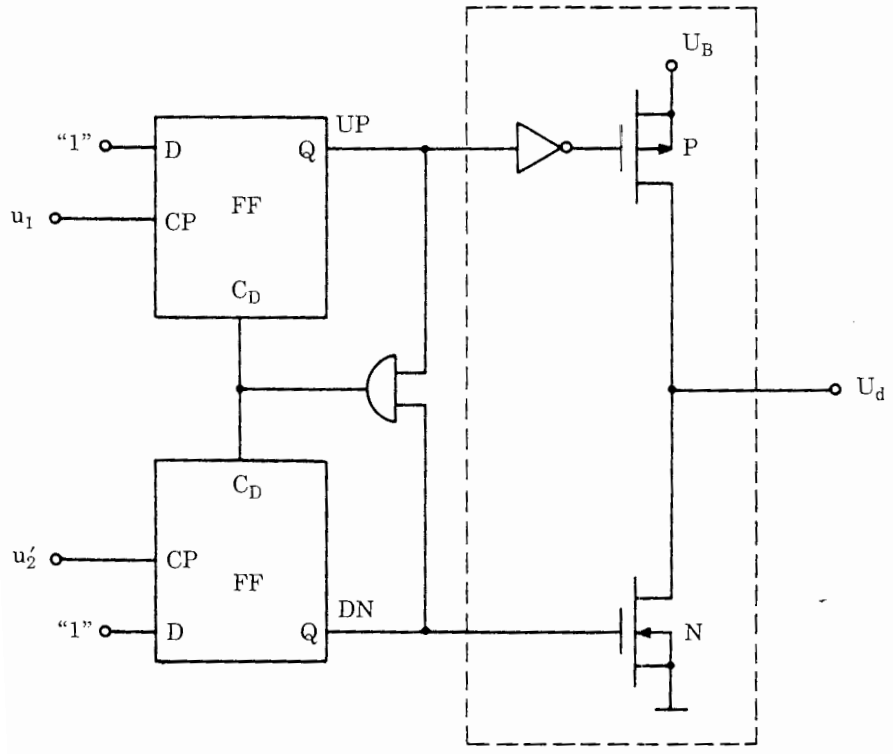


Figure 2.11 Schematic diagram of the PFD.

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 = 0 → state = –1
- UP = 0, DN = 0 → state = 0
- UP = 1, DN = 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 diagram, Fig. 2.12. (In this example we assumed that the PFD acts on the positive edges of these signals exclusively; we could have reversed the definition by saying that 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, 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 logic circuits used today generate binary signals, however, 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, i.e., is in the

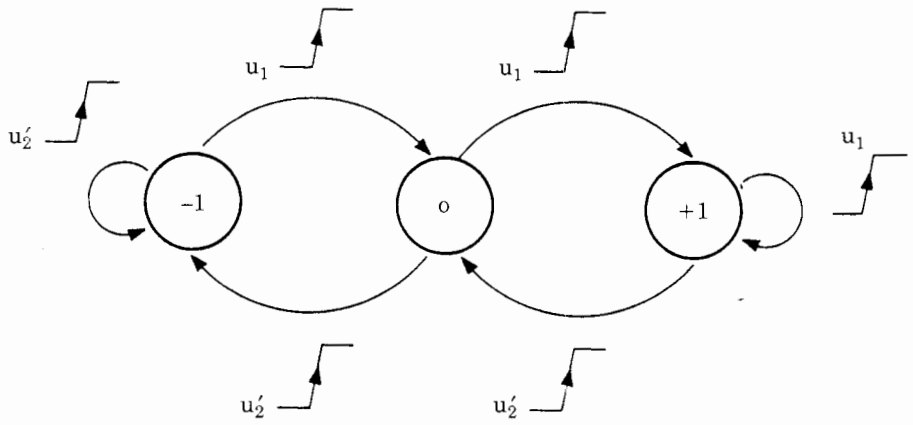


Figure 2.12 State diagram for the phase-frequency detector (PFD). This drawing shows the events causing the PFD to change its current state.

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 2.13*a* shows the (rather theoretical) case where the phase error is zero. It is assumed that 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 2.13*b* 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. 2.13*c*, the PFD toggles between states –1 and 0. It is easily seen from the waveforms in Fig. 2.13*b* and *c* that u_d becomes largest when the phase error is positive and approaches 360° (Fig. 2.13*b*) and smallest when the phase error is negative and approaches –360° (Fig. 2.13*c*). If we plot the average signal $\overline{u_d}$ versus phase error θ_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 2π or smaller than -2π. When the phase error θ_e exceeds 2π, the PFD behaves as if the phase error recycled at zero; hence the characteristic curve of the PFD becomes periodic with period 2π. An analogous consideration can be made for phase errors smaller than -2π. When the phase error is restricted to the range $-2\pi < \theta_e < 2\pi$, the average signal $\overline{u_d}$ becomes

$$\overline{u_d} = K_d \theta_e \tag{2.23}$$

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

$$K_d = \frac{U_B}{4\pi} \tag{2.24}$$

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

$$K_d = \frac{U_{\text{sat}+} - U_{\text{sat}-}}{4\pi} \tag{2.25}$$

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 that the PLL is unlocked initially. Furthermore, we make the assumption that the reference frequency ω_1 is higher that the output frequency ω_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 ω_1 is much higher than ω_2', furthermore, the PFD will be in the 1 state most of the time. When ω_1 is smaller than ω_2', however, the PFD will toggle between the states –1 and 0. When ω_1 is much lower than ω_2', the PFD will be in the –1 state most of the time. We con-

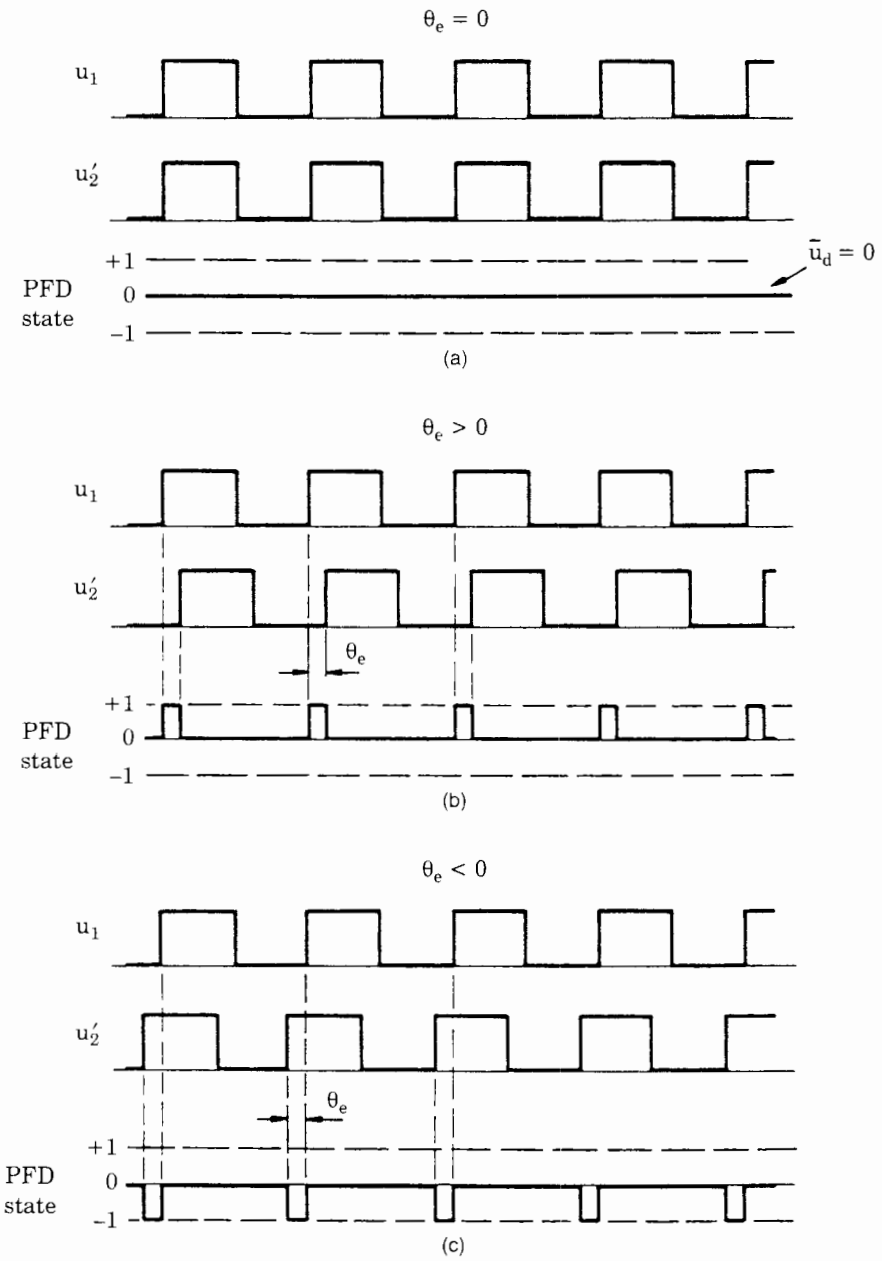


Figure 2.13 Waveforms of the signals for the PFD. (*a*) Waveforms for zero phase error. The output signal of the PFD is permanently in the 0 state (high impedance). (*b*) Waveforms for positive phase error. The PFD output signal is pulsed to the 1 state. (*c*) Waveforms for negative phase error. The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

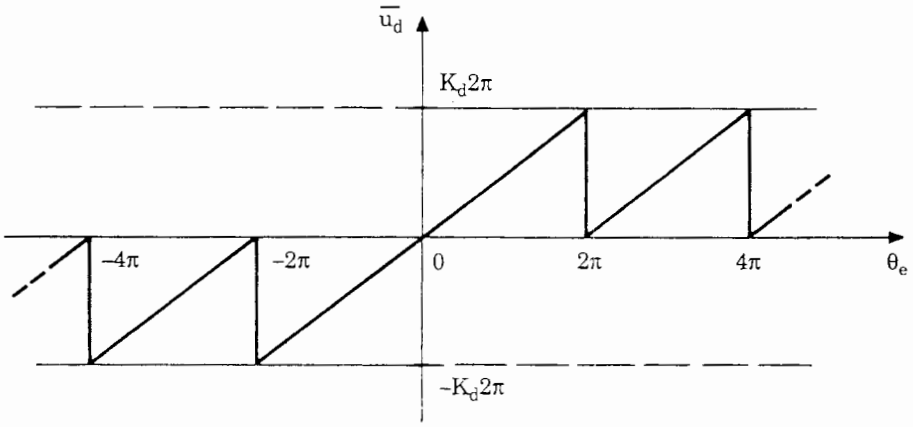


Figure 2.14 Plot of the averaged PFD output signal $\overline{u_d}$ versus phase error θ_e. $\overline{u_d}$ does not depend on the duty cycle of u_1 and u_2.

clude, 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 ω_1/ω_2' [23]; the result of this analysis is shown in Fig. 2.15. For the case $\omega_1 > \omega_2'$ the duty cycle δ is defined as the average fraction of time the PFD is in the 1 state; for $\omega_1 < \omega_2'$, δ is by definition minus the average fraction of time the PFD is in the -1 state. As expected, δ approaches 1 when $\omega_1 >> \omega_2'$ and -1 when $\omega_1 << \omega_2'$. Furthermore, δ is nearly 0.5 when ω_1 is greater than ω_2' but both frequencies are close together, and δ is nearly -0.5 when ω_1 is lower than ω_2' but both frequencies are close together. This property will greatly simplify the determination of the pull-in range (see "Pull-in Range $\Delta\omega_P$ and Pull-in Time T_P" in Sec. 2.6.2).

It must be emphasized that no such characteristic (Fig. 2.15) can be defined for the EXOR and for the JK-flipflop. Because the output signal $\overline{u_d}$ 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 that 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.

2.3.2 Loop filters (first order)

As we have seen in Sec. 2.3.1, 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 and is roughly proportional to the phase error θ_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

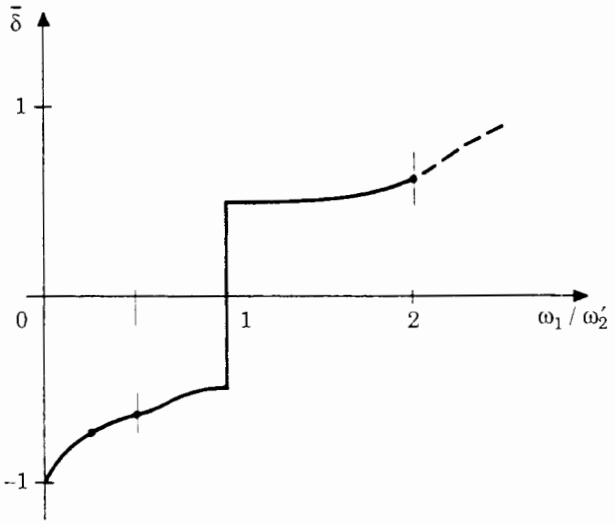


Figure 2.15 Plot of the averaged duty cycle of the PFD output signal $\overline{u_d}$ versus frequency ratio ω_1/ω_2'. This curve depicts the behavior of the PFD in the unlocked state of the DPLL.

suppress the higher, it must be a low-pass filter. In most PLL designs a first-order low-pass filter is used. Figure 2.16 lists the versions which are most frequently encountered.

Type 1: Passive lead-lag filter. Figure 2.16*a* is a passive lead-lag filter having one pole and one zero. Its transfer function $F(s)$ is given by

$$F(s) = \frac{1 + s\tau_2}{1 + s(\tau_1 + \tau_2)} \tag{2.26}$$

where $\tau_1 = R_1C$ and $\tau_2 = R_2C$. [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 (i.e., from the zero) in the transfer function in Eq. (2.26), whereas the denominator (i.e., the pole) produces the phase lag. All filters that will be used as loop filters are lead-lag filters.] The amplitude response of this filter is shown in Fig. 2.17*a*. As we will see in Sec. 2.4.2, the zero of this filter is crucial because it has a strong influence on the damping factor ζ of the PLL system.

Type 2: Active lead-lag filter. Figure 2.16*b* shows an active lead-lag filter. Its transfer function is very similar to the passive but has an additional gain term K_a, which can be chosen greater than 1. Its transfer function $F(s)$ is given by

$$F(s) = K_a \frac{1 + s\tau_2}{1 + s\tau_1} \tag{2.27}$$

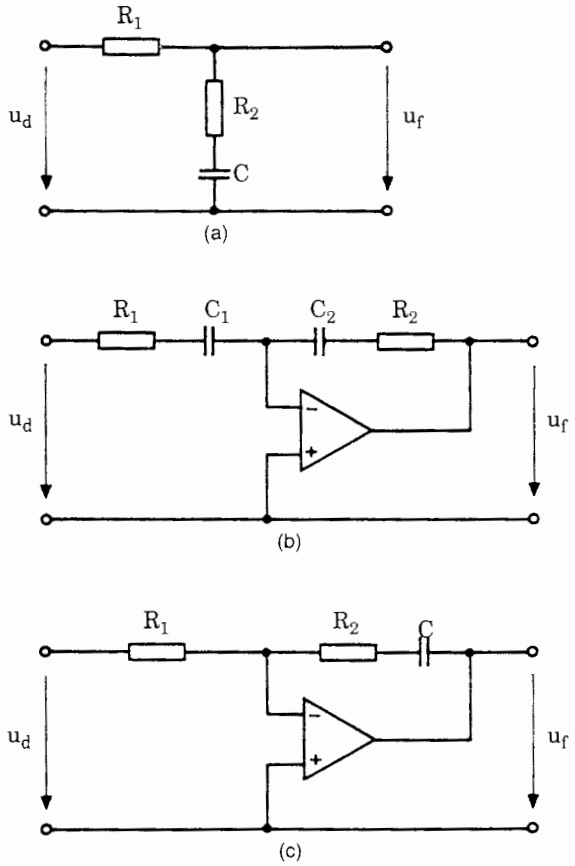


Figure 2.16 Schematic diagram of loop filters used in linear PLLs. (*a*) Passive lead-lag filter. (*b*) Active lead-lag filter. (*c*) Active PI filter.

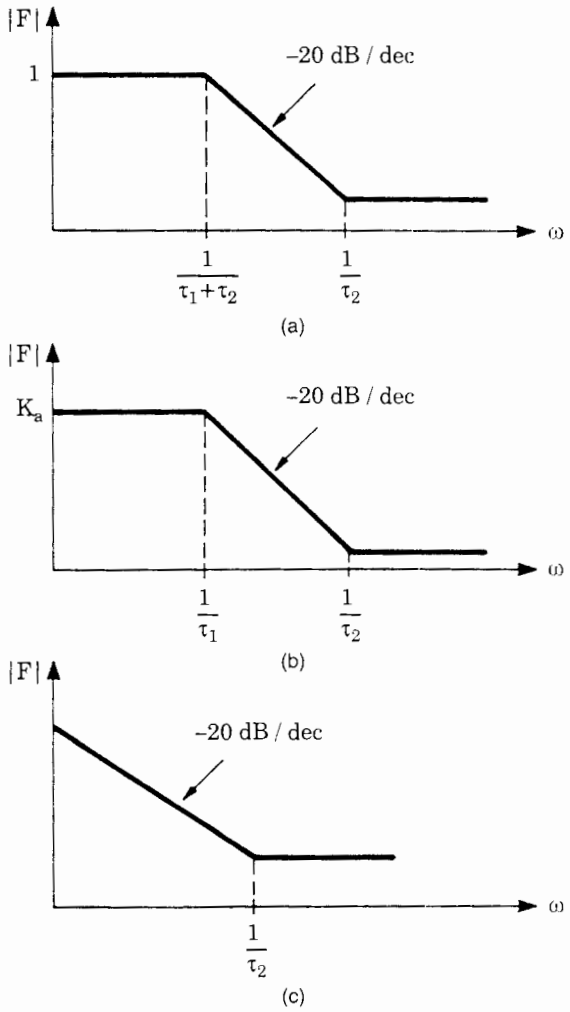


Figure 2.17 Bode diagram of the three filter types shown in Fig. 2.16. (*a*) Passive lead-lag filter. (*b*) Active lead-lag filter. (*c*) Active PI filter.

where $\tau_1 = R_1C$, $\tau_2 = R_2C$, and $K_a = C_1/C_2$. The amplitude response of the active lag filter is given in Fig. 2.17*b*.

Type 3: Active PI filter. Finally, Fig. 2.16*c* shows another active low-pass filter, which is commonly referred to as a PI filter. It is a lead-lag filter as well. The term PI is taken from control theory, where it stands for "proportional + integral" action. The transfer function of the PI filter is given by

$$F(s) = \frac{1 + s\tau_2}{s\tau_1} \tag{2.28}$$

where again $\tau_1 = R_1C$ and $\tau_2 = R_2C$. The PI filter has a pole at $s = 0$ and therefore behaves like an integrator. It has—at least theoretically—infinite gain at zero frequency. Its amplitude response is depicted in Fig. 2.17*c*.

2.3.2 Controlled oscillators

There are two types of controlled oscillators in use: voltage-controlled oscillators (VCOs) and current-controlled oscillators. They differ only in the fact that for the first the input (control) signal is a voltage signal, whereas for the second it is a current signal. Fig. 2.18 shows a simplified schematic of a VCO that 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 and $N2$ 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

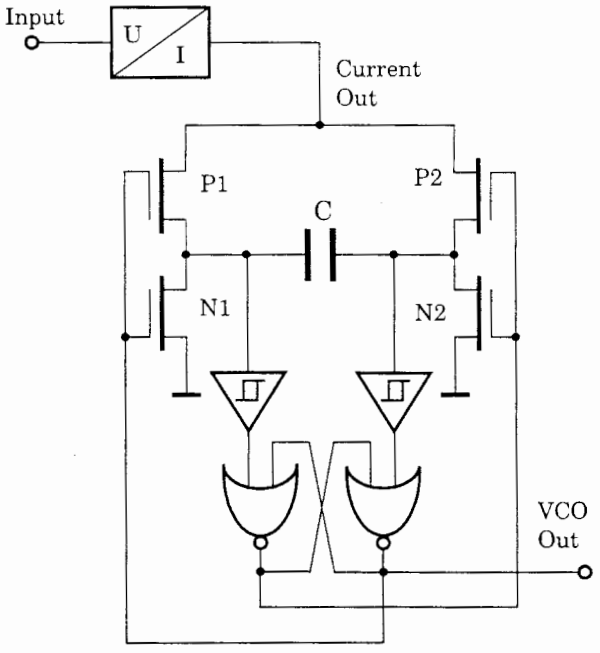


Figure 2.18 Simplified schematic of a VCO.

ramps up now, until the right Schmitt trigger switches into the H state. This process repeats infinitely. If we were to measure the voltage across the capacitor differentially, we would observe a triangular waveform.

The radian frequency ω_2 of the VCO output signal is proportional to the control signal u_f and is given by

$$\omega_2 = \omega_0 + K_0 u_f \tag{2.29}$$

K_0 is called VCO gain; its unit is rad s^{-1} V^{-1}. The unit rad is often omitted because it is a dimensionless quantity. ω_0 is the (radian) center frequency of the PLL.

Figure 2.19 shows an idealized characteristic (ω_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 $u_f = u_{fmin}$ and 2 ω_0 for $u_f = u_{fmax}$. Practical VCOs behave differently, however. First of all, most VCOs are powered from a unipolar power supply. Assuming that the supply voltage is U_{DD}, the range of u_f must be within 0 . . . U_{DD}. Real VCOs operate at their center frequency when the control signal is at half the supply voltage; i.e., $u_f = U_{DD}/2$. 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 restricted between a lower and an upper threshold. For a typical VCO implemented in CMOS technology with $U_{DD} = 5$ V, the lower treshold 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 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 that is independent of the control signal.

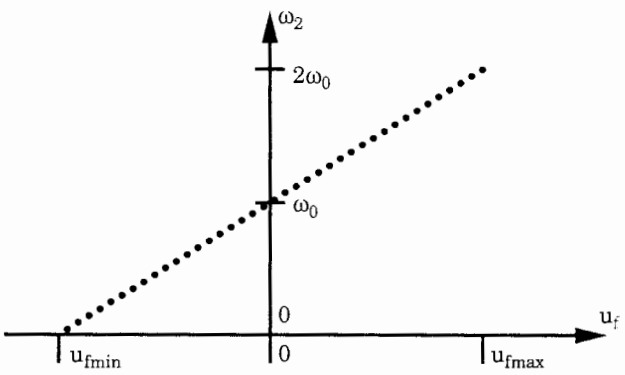


Figure 2.19 Characteristic of a VCO; ω_2 versus u_f.

The designer of a PLL system must determine two VCO parameters: the center frequency ω_0 and the VCO gain K_0. In practical VCO circuits, these parameters are set by external components, i.e., by resistors and capacitors. Figure 2.20 demonstrates how this is done in the popular PLL IC of type 74HC/HCT4046. The supply voltage is chosen as $U_{DD} = 5\,\text{V}$ in this example. The 74HC/HCT4046 uses three external components to set the parameters of the VCO, i.e., one capacitor C and two resistors R_1 and R_2. The resistors are not shown in the schematic of Fig. 2.19. Let us first consider the case where R_2 is chosen infinite; i.e., R_2 is an open circuit. The center radian frequency ω_0 of the VCO is determined by the product R_1C; refer to the solid transfer curve in Fig. 2.20. For a supply voltage of $U_{DD} = 5\,\text{V}$, 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 (see $\omega_{2\text{min}}$ and $\omega_{2\text{max}}$ 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. But we want to specify K_0 independently of center frequency ω_0. This is accomplished by adding another resistor R_2. This resistor introduces an offset. For a supply voltage of $U_{DD} = 5\,\text{V}$ this offset is approximatively

$$\Delta\omega_{\text{offset}} \approx \frac{50}{R_1 C}$$

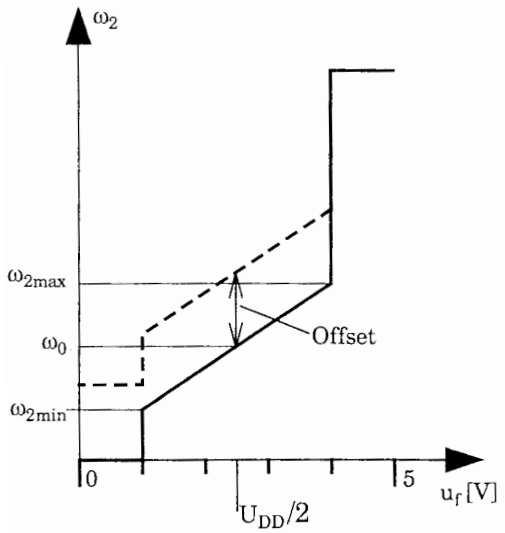


Figure 2.20 Characteristics of a typical VCO.

Using a resistor R_2 that has a finite value (e.g., 10 kΩ) shifts the transfer curve in Fig. 2.20 upward (the dashed line). Hence, by using two resistors, we are in a position to fix ω_0 and K_0 independently of each other.

Because the VCO is an analog circuit, its parameters (e.g., ω_0) are subject to parts spread, temperature drift, and aging. Most data sheets specify the temperature coefficient of ω_0, which is normally given as parts per million (ppm) per degree Centigrade.

2.3.4 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 (e.g., 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, etc. Arbitrary scaling factors (i.e., 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,[10,48] for example, and will not be discussed further here.

It is obvious that a counter can scale down by an integer number only, e.g., 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 an output frequency that is, for example, 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 rather by a technique referred to as *pulse removal*. This will be discussed in more detail in Sec. 3.2.2.

2.4 PLL Perfomance in the Locked State

If we assume that the PLL has locked and stays locked for the near future, we can develop a linear mathematical model for the system. As will be shown in this section, the mathematical model is used to calculate a phase-transfer function $H(s)$ that relates the phase θ_1 of the input signal to the phase θ_2' of the output signal (of the down-scaler):

$$H(s) = \frac{\Theta_2'(s)}{\Theta_1(s)} \tag{2.30}$$

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 θ_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. 2.4.1.

2.4.1 Mathematical model for the locked state

As derived in Sec. 2.3.1, in the locked state the output signal u_d of the phase detector can be approximated by

$$u_d \approx K_d \theta_e$$

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

$$\frac{U_d(s)}{\Theta_e(s)} = K_d \tag{2.31}$$

where s is the Laplace operator. (Appendix B provides an introduction to Laplace transforms.)

The transfer function of the loop filter has already been derived in Sec. 2.3.2 and is denoted here with $F(s)$. Let us look now at the transfer function of the VCO. As stated in Eq. (1.1), the angular frequency of the VCO is given by

$$\omega_2(t) = \omega_0 + \Delta\omega_2(t) = \omega_0 + K_d u_f(t) \tag{2.32}$$

where K_0 is called VCO gain (dimension: rad s^{-1} V^{-1}). The model of the VCO should yield the output phase θ_2, however, and not the output frequency ω_2. By definition, the phase θ_2 is given by the integral over the frequency variation $\Delta\omega_2$:

$$\theta_2(t) = \int \Delta\omega_2(t)dt = K_0 \int u_f(t)dt \tag{2.33a}$$

In the Laplace transform, integration over time corresponds to division by s, so the Laplace transform of the output phase θ_2 is given by

$$\Theta_2(s) = \frac{K_0}{s} U_f(s) \tag{2.33b}$$

The transfer function of the VCO is therefore given by

$$\frac{\Theta_2(s)}{U_f(s)} = \frac{K_0}{s} \tag{2.34}$$

For phase signals the VCO simply represents an integrator.

Last we must also know the mathematical model of the (optional) divide-by-N counter (see 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, Fig. 2.21. On the basis of this model we are going to derive a number of transfer functions and related parameters (see Sec. 2.4.2).

2.4.2 Definition of transfer functions

The model in Fig. 2.21 enables us to analyze the tracking performance of the PLL, i.e., the ability of the system to maintain phase tracking when excited by phase steps, frequency steps, or other excitation signals.

From the model, the phase-transfer function $H(s)$ is computed. We get

$$H(s) = \frac{\Theta_2'(s)}{\Theta_1(s)} = \frac{K_0 K_d F(s)/N}{s + K_0 K_d F(s)/N} \tag{2.35}$$

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

$$H_e(s) = \frac{\Theta_e(s)}{\Theta_1(s)} = \frac{s}{s + K_0 K_d F(s)/N} \tag{2.36}$$

$H_e(s)$ relates phase error θ_e to the input phase θ_1. Between $H_e(s)$ and $H(s)$ we have the simple relation

$$H_e(s) = 1 - H(s) \tag{2.37}$$

To analyze the phase-transfer function we have to insert the loop filter transfer function [Eqs. (2.26) to (2.28)] into Eq. (2.35). For the three different loop filters (Fig. 2.16) we get

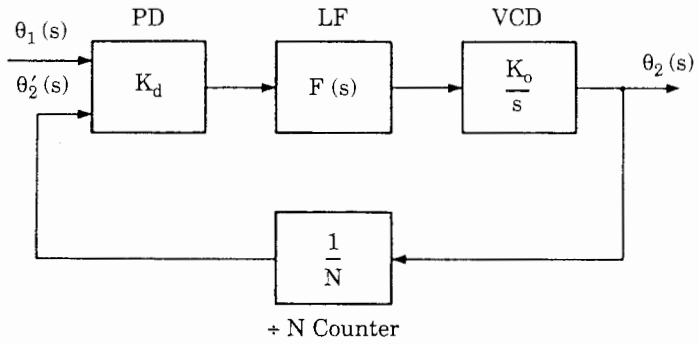


Figure 2.21 Mathematical model for the locked state of the PLL.

- For the passive lead-lag filter

$$H(s) = \frac{\dfrac{K_0 K_d}{N}\dfrac{1+s\tau_2}{\tau_1+\tau_2}}{s^2 + s\dfrac{1+K_0K_d\tau_2/N}{\tau_1+\tau_2} + \dfrac{K_0K_d/N}{\tau_1+\tau_2}} \tag{2.38}$$

- For the active lead-lag filter

$$H(s) = \frac{\dfrac{K_0 K_d K_a}{N}\dfrac{1+s\tau_2}{\tau_1}}{s^2 + s\dfrac{1+K_0K_dK_a\tau_2/N}{\tau_1} + \dfrac{K_0K_dK_a/N}{\tau_1}} \tag{2.39}$$

- For the active PI filter

$$H(s) = \frac{\dfrac{K_0 K_d}{N}\dfrac{1+s\tau_2}{\tau_1}}{s^2 + s\dfrac{1+K_0K_d\tau_2/N}{\tau_1} + \dfrac{K_0K_d/N}{\tau_1+\tau_2}} \tag{2.40}$$

In circuit and control theory it is common practice to write the denominator of the transfer function in the so-called normalized form[3]

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

where ω_n is the natural frequency and ζ is the damping factor. The denominator of Eqs. (2.38) to (2.40) will take this form if the following substitutions are made:

- For the passive lead-lag filter

$$\omega_n = \sqrt{\frac{K_0K_d}{N(\tau_1+\tau_2)}},\quad \zeta = \frac{\omega_n}{2}\left(\tau_2 + \frac{N}{K_0K_d}\right) \tag{2.41}$$

- For the active lead-lag filter

$$\omega_n = \sqrt{\frac{K_0K_dK_a}{N(\tau_1+\tau_2)}},\quad \zeta = \frac{\omega_n}{2}\left(\tau_2 + \frac{N}{K_0K_dK_a}\right) \tag{2.42}$$

- For the active PI filter

$$\omega_n = \sqrt{\frac{K_0K_d}{N\tau_1}},\quad \zeta = \frac{\omega_n}{2}\tau_2 \tag{2.43}$$

(The natural frequency ω_n must never be confused with the center frequency ω_0 of the PLL.) Inserting these substitutions into Eqs. (2.38) to (2.40), we get the following phase-transfer functions:

- For the passive lead-lag filter

$$H(s) = \frac{s\omega_n\left(2\zeta - \dfrac{\omega_n}{K_0 K_d / N}\right)}{s^2 + 2s\zeta\omega_n + \omega_n^2} \tag{2.44}$$

- For the active lead-lag filter

$$H(s) = \frac{s\omega_n\left(2\zeta - \dfrac{\omega_n}{K_0 K_d K_a / N}\right)}{s^2 + 2s\zeta\omega_n + \omega_n^2} \tag{2.45}$$

- For the active PI filter

$$H(s) = \frac{2s\zeta\omega_n + \omega_n^2}{s^2 + 2s\zeta\omega_n + \omega_n^2} \tag{2.46}$$

Aside from the parameters ω_n and ζ, only the parameters K_d, K_0, K_a, and N appear in Eqs. (2.44 to 2.46). The term $K_d K_0/N$ in Eqs. (2.44) and (2.46) is called *loop gain* and has the dimension of angular frequency (rad s^{-1}). In Eq. (2.45), the term $K_d K_0 K_a/N$ is called *loop gain*. If the condition

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

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. (2.44) to (2.46) become approximately identical and read

$$H(s) \approx \frac{2s\zeta\omega_n + \omega_n^2}{s^2 + 2s\zeta\omega_n + \omega_n^2} \tag{2.47}$$

for all filters shown in Fig. 2.16. Similarly, assuming a high-gain loop, we get for the error-transfer function H_e(s) for all three filter types the approximate expression

$$H_e(s) \approx \frac{s^2}{s^2 + 2s\zeta\omega_n + \omega_n^2} \tag{2.48}$$

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. (2.47) and by plotting the magnitude (absolute value) $|H(\omega)|$ as a function of angular frequency ω (Fig. 2.22). Both scales are usually logarithmic. The frequency scale is further nor-

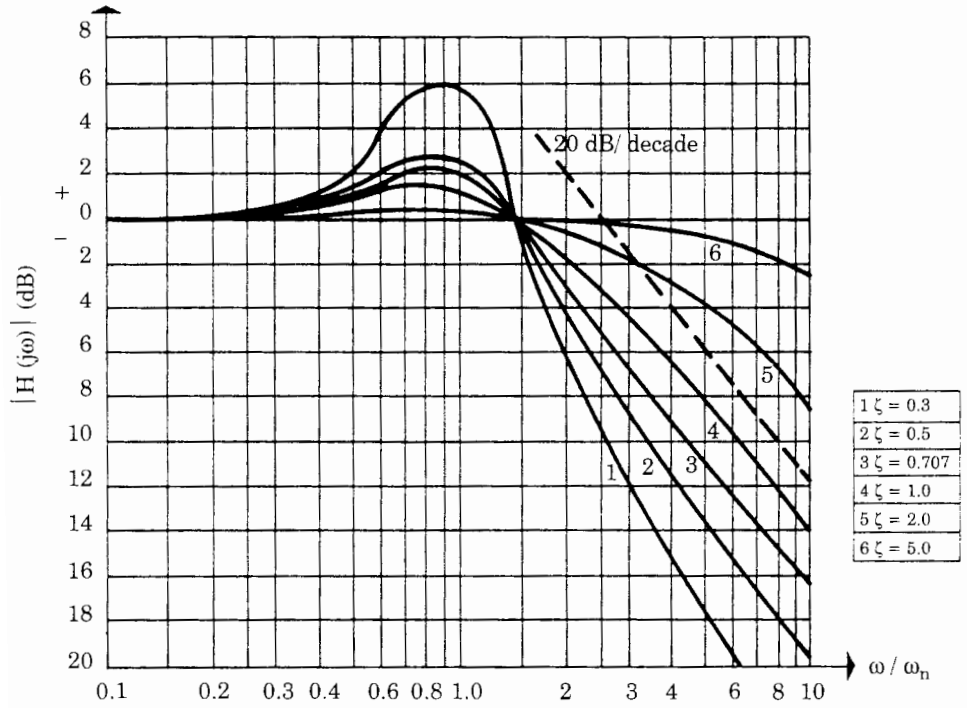


Figure 2.22 Bode diagram of the phase-transfer function $H(\omega)$. (*Adapted from Gardner*[1] *with permission.*)

malized to the natural frequency ω_n. Thus the graph is valid for every second-order PLL system.

We can see from Fig. 2.22 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 ω_n. This means that 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 roughly between zero and ω_n. The damping factor ζ has an important influence on the dynamic performance of the PLL. For $\zeta = 1$, the system is critically damped.

If ζ 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 ζ is made considerably larger than unity, the transfer function flattens out, and the dynamic response becomes sluggish.

A Bode plot of $H_e(s)$ is shown in Fig. 2.23. The value of 0.707 has been chosen for ζ. The diagram shows that, for modulation frequencies smaller than the natural frequency ω_n, the phase error remains relatively small. For larger frequencies, however, the phase error θ_e becomes as large as the reference phase θ_1, which means that the PLL is no longer able to maintain phase tracking.

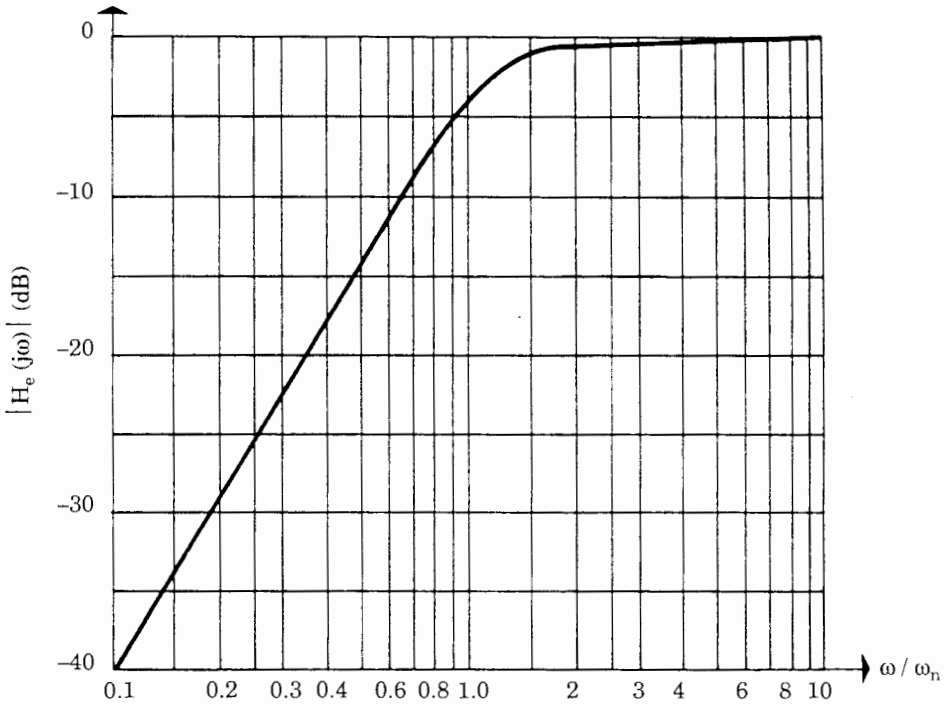


Figure 2.23 Bode diagram of the error-transfer function $H_e(\omega)$.

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

$$\omega_{3dB} = \omega_n \left[1 + 2\zeta^2 + \sqrt{(1+2\zeta^2)^2 + 1} \right]^{1/2} \tag{2.49}$$

For a damping factor $\zeta = 0.7$, ω_{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. 2.24). 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 θ_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 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 large phase errors θ_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 that the PLL is initially locked. If the PLL is initially unlocked, however, the phase error θ_e can take on arbitrarily

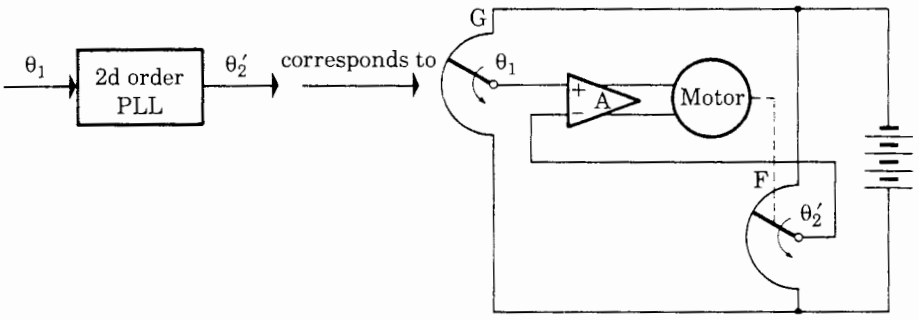


Figure 2.24 Simple electromechanical analogy of the linearized second-order PLL. In this servo system, the angles θ_1 and θ_2' correspond to the phases θ_1 and θ_2', respectively, of the PLL system.

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. 2.6.1.

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. 2.4.3.

2.4.3 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 PLLs 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. 2.2*a*. In this case the phase signal $\theta_1(t)$ is a step function,

$$\theta_1(t) = u(t) \cdot \Delta\Phi \tag{2.50}$$

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 get therefore

$$\Theta_1(s) = \Delta\Phi/s \tag{2.51}$$

The phase error θ_e is obtained from

$$\Theta_e(s) = H_e(s)\Theta_1(s) = H_e(s) \cdot \Delta\Phi/s \tag{2.52}$$

Inserting Eq. (2.48) into Eq. (2.52) yields

$$\Theta_e(s) = \frac{\Delta\Phi}{s} \frac{s^2}{s^2 + 2s\zeta\omega_n + \omega_n^2} \tag{2.53}$$

Applying the inverse Laplace transform to Eq. (2.53), we get the phase-error functions $\theta_e(t)$ shown in Fig. 2.25. 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 ζ. For $t \to (\infty)$ the phase error $\theta_e(\infty)$ approaches zero. This can also be shown by the final value theorem of the Laplace transform,[3] which reads

$$\theta_e(\infty) = \lim_{s \to 0} s\Theta_e(s) = 0 \tag{2.54}$$

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

$$\omega_1(t) = \omega_0' + \Delta\omega \cdot u(t) \tag{2.55}$$

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

$$\theta_1(t) = \Delta\omega \cdot t \tag{2.56}$$

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

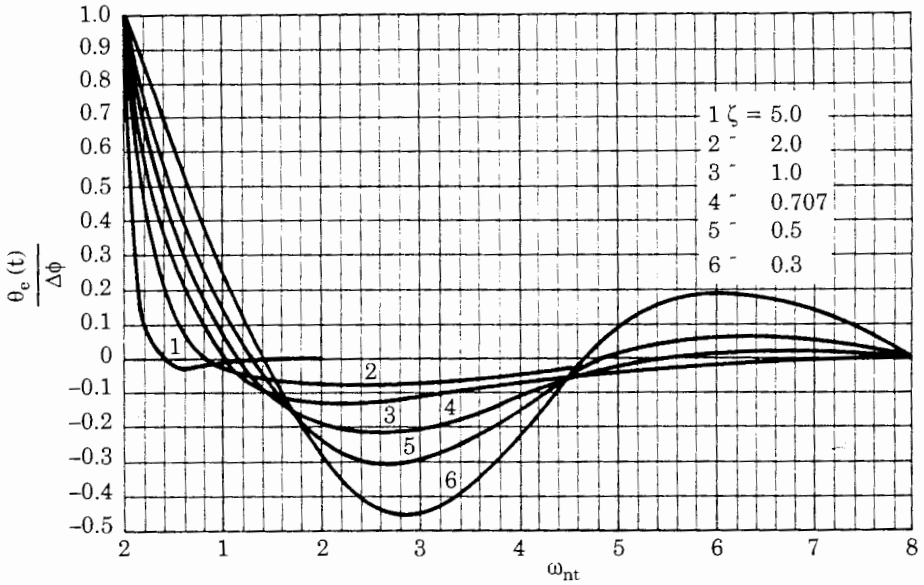


Figure 2.25 Transient response of a linear second-order PLL to a phase step $\Delta\Phi$ applied at $t = 0$. The PLL is assumed to be a high-gain loop. (*Adapted from Gardner*[1] *with permission.*)

$$\Theta_1(s) = \Delta\omega/s^2 \tag{2.57}$$

Performing the same calculation as above, we get for the phase error

$$\Theta_e(s) = \frac{\Delta\omega}{s^2}\frac{s^2}{s^2 + 2s\zeta\omega_n + \omega_n^2} \tag{2.58}$$

Applying the inverse Laplace transform to Eq. (2.58), we get the phase error curves shown in Fig. 2.26. If we again apply the final value theorem to Eq. (2.58), it turns out that the phase error approaches 0 when $t \to \infty$. This is true, however, only for high-gain loops. For low-gain loops, the numerator of Eq. (2.58) 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 ω_1 is given by

$$\omega_1(t) = \omega_0{}' + \Delta\dot{\omega} \cdot t \tag{2.59}$$

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

$$\theta_1(t) = \Delta\dot{\omega}\frac{t^2}{2} \tag{2.60}$$

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

$$\Theta_1(s) = \frac{\Delta\dot{\omega}}{s^3} \tag{2.61}$$

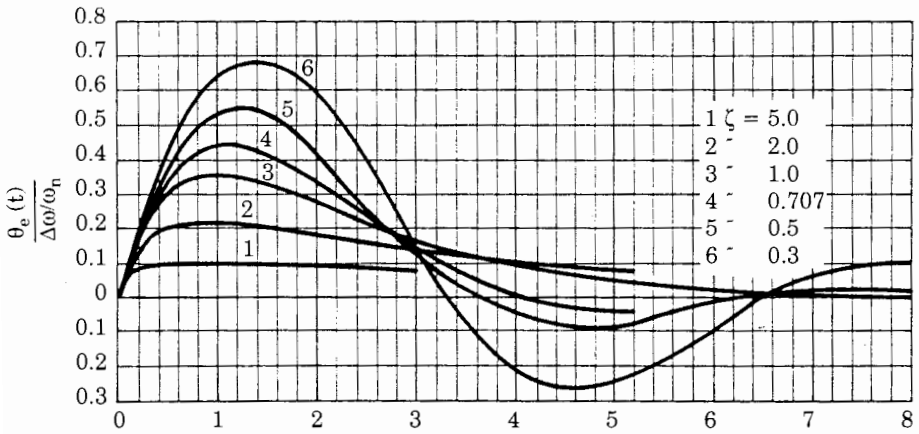


Figure 2.26 Transient response of a linear second-order PLL to a frequency step $\Delta\omega$ applied to its reference input at $t = 0$. (*Adapted from Gardner*[1] *with permission.*)

Assuming a high-gain loop again and applying the final value theorem of the Laplace transform, the final phase error $\theta_e(\infty)$ is

$$\theta_e(\infty) = \lim_{s\to 0} sH_e(s)\Theta_1(s) = \frac{\Delta\dot{\omega}}{\omega_n^2} \tag{2.62}$$

Now we remember that our linear model is valid for small phase errors only. Because for large phase errors the output signal of the phase detector is not proportional to the phase error, but to the sine of the phase error, the last equation can be shown to read actually[1]

$$\sin\theta_e(\infty) = \frac{\Delta\dot{\omega}}{\omega_n^2} \tag{2.63}$$

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

$$\Delta\dot{\omega}_{\max} = \omega_n^2 \tag{2.64}$$

This result has two consequences:

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

Practical experience with PLLs has shown that Eq. (2.64) presents a theoretical limit which 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 ω_n^2. A more practical design limit for $\Delta\dot{\omega}_{\max}$ is considered to be[1]

$$\Delta\dot{\omega}_{\max} = \frac{\omega_n^2}{2} \tag{2.65}$$

2.4.4 Steady-state error of the PLL

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. (2.48). It turns out that the steady-state error depends largely on the number of "integrators" that are present in the control system, i.e., on the number of poles at $s = 0$ of the open-

loop transfer function. Using Eq. (2.36) and the final value theorem of the Laplace transform [see App. B, Eq. (B.19)], we can write $\theta_e(\infty)$ as

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

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. (2.28), $M = 1$. 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

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

Let us calculate now 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 $\Delta\Phi$ is applied to the reference input, we have (cf. Sec. 2.4.3)

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

Hence the steady-state error becomes

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

This quantity becomes 0 for any M, i.e., 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. 2.4.3, and for its Laplace transform we get

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

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

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

This becomes 0 only if M is greater than or equal to 1, i.e., if the loop filter has at least one pole at $s = 0$. In other words, 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. 2.4.3, 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

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

Here the steady-state error approaches zero only if the loop filter has at least 2 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 90^0 at higher frequencies, the overall phase shift can approach 270^0, which means that the system can get unstable. To get a stable loop, the poles must be compensated for by zeros; i.e., the loop filter must have at least one zero, which must be correctly placed, of course.[3]

2.5 The Order of the PLL System

2.5.1 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 by 1 higher than the order of the loop filter. If the loop filter is omitted, i.e., 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. 2.5.2. In PLL applications, first-order systems are rarely used, because they offer little noise suppression (more about noise in later sections).

Because higher-order loop filters offer better noise cancellation, loop filters of order 2 and more 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. 4.

2.5.2 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. (2.35) becomes 1, and for the phase transfer function we obtain

$$H(s) = \frac{1}{1 + \dfrac{sN}{K_0 K_d}} \tag{2.66}$$

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. 2.6.2, 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 ω_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. Because of its high bandwidth, however, the first-order PLL does not suppress noise superimposed on the input signal. Because this is an undesirable property in most PLL applications, the first-order PLL is rarely utilized here. First-order PLLs do really exist in applications where noise is not a primary concern. In Chap. 6 we will deal with a first-order all-digital PLL system that is frequently used in FSK modems.

2.6 PLL Performance in the Unlocked State

As we have seen in Sec. 2.4, the linear model of the LPLL 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 are going to develop a mathematical model that is valid in the unlocked state of the PLL. On the basis of that model, we will develop a mechanical analogy that is much simpler to analyze and will help us to derive the relevant parameters that describe the acquisition and lockout processes of PLLs.

The 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?

2.6.1 Mathematical model for the unlocked state

The mathematical model depends somewhat on the types of phase detectors and loop filters that are used in a particular PLL configuration. For the following analysis, we assume that 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[4]

$$\ddot{\theta}_e + \dot{\theta}_e \frac{1 + K_0 K_d \tau_2 \cos\theta_e / N}{\tau_1 + \tau_2} + \frac{K_0 K_d / N}{\tau_1 + \tau_2} \sin\theta_e = \ddot{\theta}_1 + \dot{\theta}_1 \frac{1}{\tau_1 + \tau_2} \tag{2.67}$$

This equation can be simplified. First, the substitutions of Eq. (2.41) are made for τ_1 and τ_2. Next, in most practical cases the inequality

$$\frac{1}{\tau_2} << K_0 K_d / N \tag{2.68}$$

holds. This leads to the simplified differential equation

$$\ddot{\theta}_e + 2\zeta\omega_n \dot{\theta}_e \cos\theta_e + \omega_n^2 \sin\theta_e = \ddot{\theta}_1 + \dot{\theta}_1 \frac{\omega_n^2}{K_0 K_d / N} \tag{2.69}$$

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. (2.69) is almost identical to the differential equation of a somewhat special mathematical pendulum, as shown in Fig. 2.27. A beam

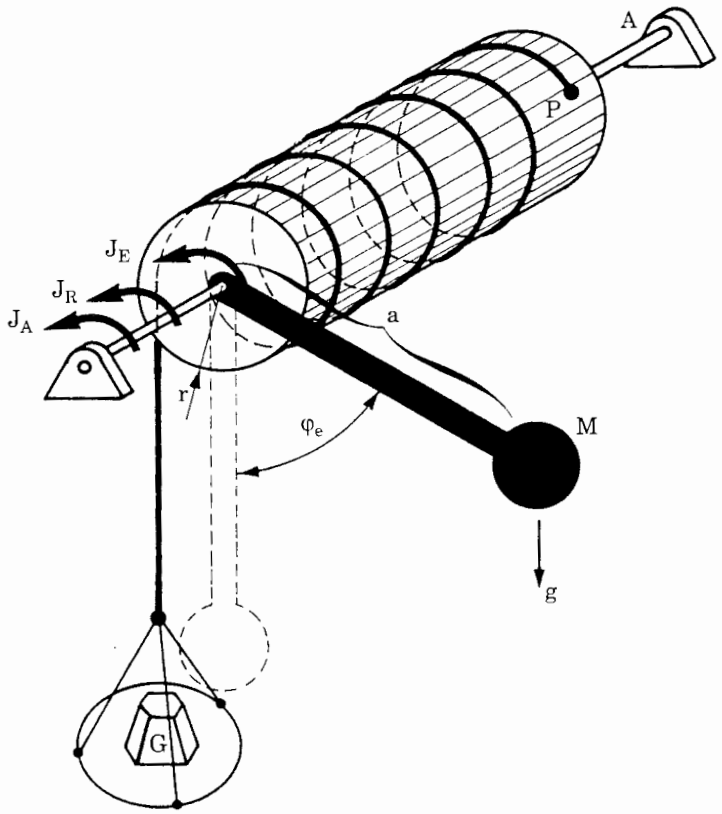


Figure 2.27 Mechanical analogy illustrating the linear and the nonlinear performance of the PLL. (Symbols defined in text.)

having a mass M is rigidly fixed to the shaft of a cylinder that 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 φ_e. The dynamic response of the pendulum can be calculated by Newton's third law,

$$T\ddot{\varphi}_e = \sum_i J_i \tag{2.70}$$

where T is the moment of inertia of the pendulum plus cylinder, φ_e is the angle of deflection, and J_i is a driving torque. Three different torques can be identified in the mechanical system of Fig. 2.27:

1. 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.
2. A friction torque J_R, which is assumed to be proportional to the angular velocity (viscous friction); $J_R = -\rho\dot{\varphi}_e$, where ρ is the coefficient of friction.
3. 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. (2.70) yields

$$\ddot{\varphi}_e + \frac{\rho}{T}\dot{\varphi}_e + \frac{Mag}{T}\sin\varphi_e = \frac{r}{T}G(t) \tag{2.71}$$

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

$$\omega_n' = \left(\frac{Mag}{T}\right)^{1/2} \tag{2.72}$$

$$\zeta' = \frac{\rho}{2\sqrt{MagT}}$$

Eq. (2.71) is converted into

$$\ddot{\varphi}_e + 2\zeta'\omega_n'\dot{\varphi}_e + \omega_n'^2 \sin\varphi_e = \frac{r}{T}G(t) \approx G(t) \tag{2.73}$$

This nonlinear differential equation for the deflection angle φ_e looks very much like the nonlinear differential equation of the PLL according to Eq. (2.69).

There is a slight difference in the second term, 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. 2.27 would be an accurate analogy of the PLL only if the friction varied with the cosine of the deflection angle. This would be true if the damping factor ζ' were not a constant but would vary 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 $\zeta' \approx \cos\varphi_e$ 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 ζ' varies with the cosine of φ_e. The mathematical pendulum is therefore a reasonable approximation for the PLL.

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

1. The phase error θ_e of the PLL corresponds to the angle of deflection φ_e of the pendulum.
2. The natural frequency ω_n of the PLL corresponds to the natural (or resonant) frequency of the pendulum ω_n'.
3. The damping factor ζ of the PLL corresponds to the damping factor ζ' of the pendulum, which results from viscous friction.
4. The weight G on the platform corresponds to a reference phase disturbance according to the relation

$$\ddot{\theta}_1 + \dot{\theta}_1 \frac{\omega_n^2}{K_0 K_d / N} \sim G(t) \tag{2.74}$$

Let us now see what is the physical meaning of the term $\ddot{\theta}_1 + \dot{\theta}_1 \omega_n^2 / (K_0 K_d / N)$ in Eq. (2.74). We assume first 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 $\Delta\dot{\omega}(t)$:

$$G(t) \sim \Delta\dot{\omega} + \frac{\omega_n^2}{K_0 K_d / N} \Delta\omega \tag{2.75}$$

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 have to place the corresponding weight $G(t)$ given by Eq. (2.75) 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 us 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 ω_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. (2.75) 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

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

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 θ_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°. 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 2π. 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

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

and is plotted in Fig. 2.28. What will now be the weight $G(t)$ required to simulate this condition? As also shown in Fig. 2.28, 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 a narrow and steep impulse, the stroke should be *elastic*. If this is so, 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 to be dropped onto the platform, we will observe a situation where

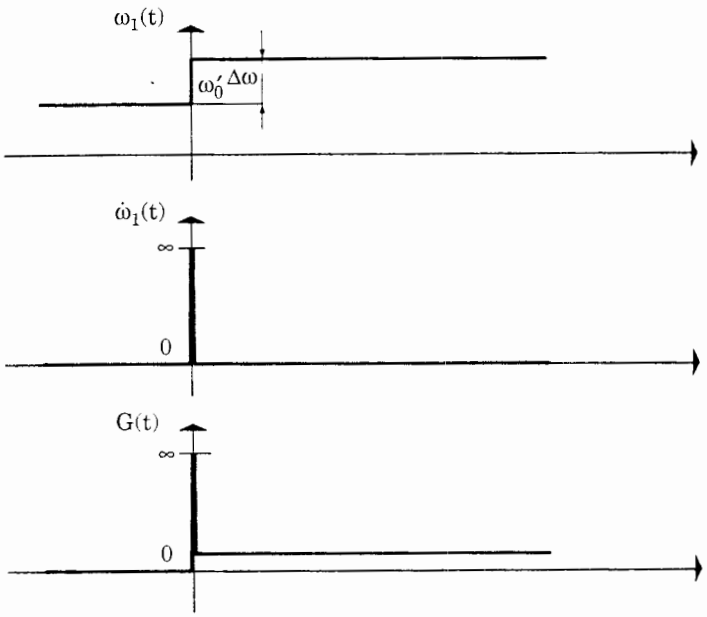


Figure 2.28 Weight function $G(t)$ required to simulate a frequency step applied to the reference input of the PLL (ω_1 = angular frequency of the reference signal; $\Delta\omega$ = frequency step applied at $t = 0$).

the peak deflection exceeds 90°, but not 180°, and the final deflection is less than 90°. We thus conclude that a linear PLL can operate stably when the phase error θ_e momentarily exceeds the value of 90°. If the weight dropped onto the platform is increased ever further, the peak deflection will exceed 180°. 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 have to define another critical frequency offset, i.e., 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.

There is a third way 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 that was generated in the last example by dropping a weight onto the platform. As has been shown in Sec. 2.4.3, the rate of change of the reference frequency must always be smaller than ω_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:

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

Whenever a PLL has lost tracking because one of these conditions has not been fulfilled, the question arises 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 buildup 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 ω_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 "Pull-in Range $\Delta\omega_P$ and Pull-in Time T_P" in Sec. 2.6.2.

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*.

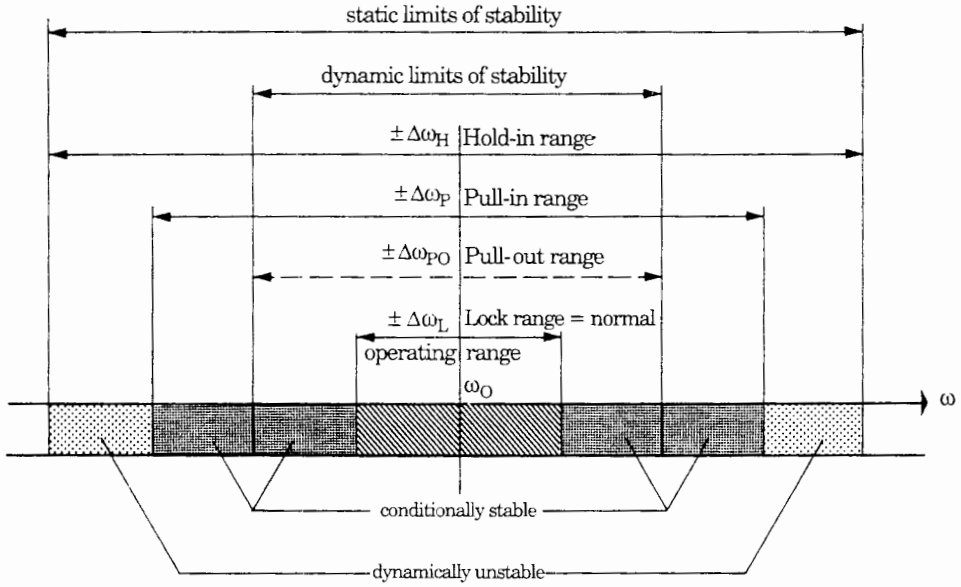


Figure 2.29 Scope of the static and dynamic limits of stability of a linear second-order PLL.

The mechanical model has shown that there are four key parameters specifying the frequency range in which the PLL can be operated. Figure 2.29 is a graphical representation of these parameters. The four key parameters can be summarized as follows:

1. 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.
2. 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.
3. 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.
4. The lock range $\Delta\omega_L$. This is the frequency range within which a PLL locks within 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. 2.6.2. The quantitative relationships among these four parameters are plotted in Fig. 2.29 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.[1]

2.6.2 Key parameters of the PLL

In Sec. 2.6.1 it has been 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 $\Delta\omega_H$

In addition we will define parameters relating to the time required for the PLL to get locked:

- The lock time T_L. This is 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, i.e., the time constants τ_1 and τ_2 of the loop filter; the gain factors K_d, 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 types of phase detector and loop filter used. To make the design easier, the results will be shown in tables. Moreover, we will demonstrate in Chap. 5 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 lock statically. As we have seen in Sec. 2.6.1, 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. 2.3.1, 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°. 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°, and for the PFD this value is even 360°. 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°. At the limit of the hold range the input frequency ω_1 is given by

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

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

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

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. (2.48):

$$\Theta_e(s) = \Theta_1(s)H_e(s) = \frac{\Delta\omega_H}{s^2}\frac{s}{s + K_0K_d F(s)/N}$$

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

$$\theta_e(\infty) = \lim_{s\to 0} s\Theta_e(s) = \frac{\Delta\omega_H}{K_0K_d F(s)/N} \tag{2.76}$$

Remember that the PLL network was linearized when the Laplace transform was introduced. Consequently Eq. (2.76) is valid for small values of θ_e only. For greater values of phase error we would have to write

$$\sin\theta_e(\infty) = \lim_{s\to 0} s\Theta_e(s) = \frac{\Delta\omega_H}{K_0K_d F(s)/N}$$

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

$$\Delta\omega_H = K_0K_d F(s)/N$$

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 lead-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

$$\begin{aligned} \Delta\omega_H &= K_0 K_d / N && \text{(passive lead-lag)} \\ &= K_0 K_d K_a / N && \text{(active lead-lag)} \\ &= \infty && \text{(active PI)} \end{aligned} \tag{2.77}$$

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 $\overline{u_d} = K_d\theta_e$ is valid for all values of θ_e in the range $-\pi/2 < \theta_e < \pi/2$. Hence we get from Eq. (2.76)

$$\Delta\omega_H = \frac{K_0 K_d F(0)\pi/2}{N} \tag{2.78}$$

Note again that $F(0)$—the loop filter gain at dc—depends on the type of loop filter chosen, as desribed above.

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

$$\Delta\omega_H = \frac{K_0 K_d F(0)\pi}{N} \tag{2.79}$$

Phase detector type 4. When the PFD is used, the average phase detector output $\overline{u_d}$ is proportional to the phase error over the range $-2\pi < \theta_e < 2\pi$. Hence we get for the hold range

$$\Delta\omega_H = \frac{K_0 K_d F(0)2\pi}{N} \tag{2.80}$$

Lock range $\Delta\omega_L$ and lock time T_L. As in the previous section, we analyze the lock range 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 is initially not locked and that the reference frequency is $\omega_1 = \omega_0' + \Delta\omega$. The reference signal of the PLL is then given by

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

and the output signal by

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

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 (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 (Fig. 2.1) is therefore given by $K_d K_0 |F(\Delta\omega)|/N$. In Fig. 2.30, the frequency ω_2' of the VCO is plotted against time for two cases. In Fig. 2.30*a* the peak frequency deviation is less than the offset $\Delta\omega$ between the reference frequency ω_1 and the down-scaled frequency ω_0'. Hence a lock-in process will not take place, at least not instantaneously.

Figure 2.30*b* shows a special case, however, where the peak frequency deviation is just as large as the frequency offset $\Delta\omega$. The frequency ω_2' of the VCO output signal develops as shown by the solid line. When the frequency deviation is at its largest, ω_2' exactly meets the value of the reference frequency ω_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 the discussion of lock range in the following subsection 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

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

For the lock range we obtain

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

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:

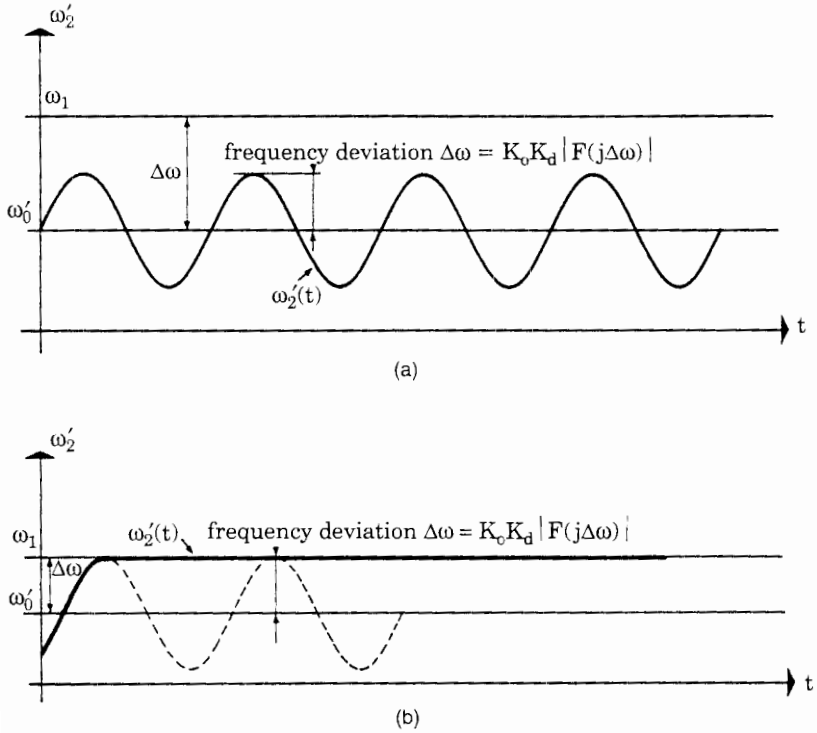


Figure 2.30 Lock-in process. (*a*) The peak frequency deviation is smaller than the offset $\Delta\omega$; therefore a fast lock-in process cannot take place. (*b*) The peak frequency deviation is exactly as large as the actual frequency offset; thus the PLL will become locked after a very short time. *Note*: in this example the divider ratio is assumed to be $N = 1$ (no down-scaler used).

$$F(\Delta\omega_L) \approx \frac{\tau_2}{\tau_1 + \tau_2} \qquad \text{for the passive lead-lag filter}$$

$$F(\Delta\omega_L) \approx K_a \frac{\tau_2}{\tau_1} \qquad \text{for the active lead-lag filter}$$

$$F(\Delta\omega_L) \approx \frac{\tau_2}{\tau_1} \qquad \text{for the active PI filter}$$

Moreover, τ_2 is normally much smaller than τ_1, so we can use the simplified relationships

$$F(\Delta\omega_L) \approx \tau_2/\tau_1 \qquad \text{for the passive lead-lag filter}$$
$$F(\Delta\omega_L) \approx K_a\,\tau_2/\tau_1 \qquad \text{for the active lead-lag filter}$$
$$F(\Delta\omega_L) \approx \tau_2/\tau_1 \qquad \text{for the active PI filter}$$

Making use of the substitutions Eq. (2.41) and assuming high-gain loops, we get

$$\Delta\omega_L \approx 2\zeta\omega_n \tag{2.81}$$

for all types of loop filters in Fig. 2.16.

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 ω_n. As Fig. 2.26 shows, the transients die out in about one cycle of this oscillation; hence it is reasonable to state that the lock-in time is

$$T_L \approx \frac{2\pi}{\omega_n} \tag{2.82}$$

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 that the PLL is initially out of lock and that both signals u_1 and u_2' are symmetrical square waves. The reference frequency ω_1 is offset from the center frequency ω_0' by $\Delta\omega$. Then for u_1 and u_2' we have

$$u_1(t) = U_{10} \text{ rect } (\omega_0' t + \Delta\omega \cdot t)$$
$$u_2(t) = U_{20} \text{ rect } (\omega_0' t)$$

where U_{10} and U_{20} are the amplitudes of the square wave signals. The phases θ_1 and θ_2' are then given by

$$\theta_1(t) = \omega_0' t + \Delta\omega \cdot t$$
$$\theta_2(t) = \omega_0' t$$

The phase error θ_e therefore is

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

which is a ramp function. Checking Fig. 2.7 again we note that 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

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

where tri stands for *triangular waveform*. This signal is depicted in the upper trace of Fig. 2.31. The average output signal of the loop filter is now given by

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

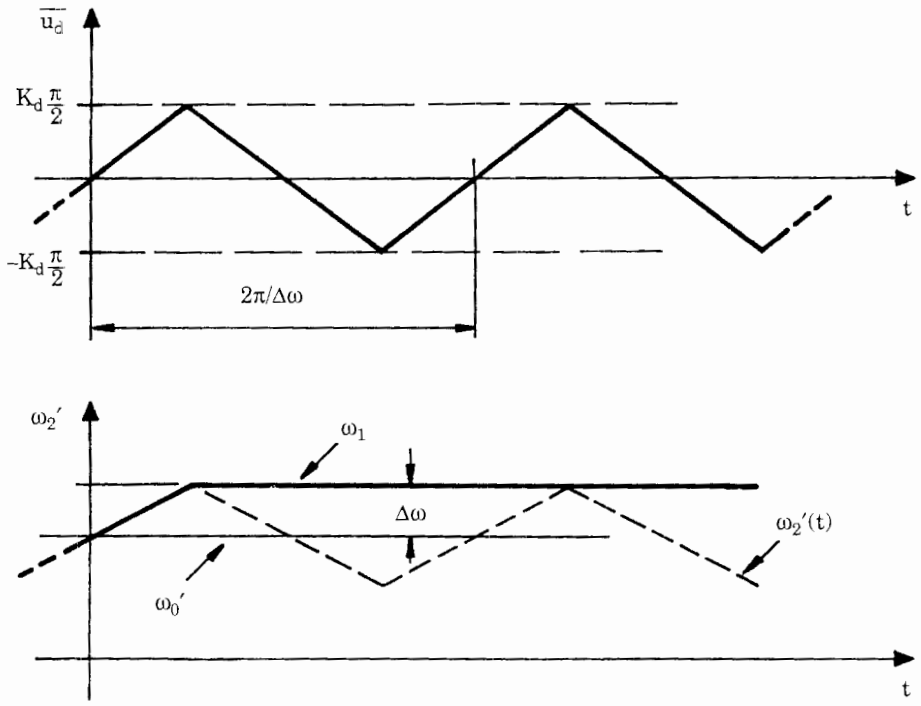


Figure 2.31 Waveforms of the average phase detector output signal $\overline{u_d}(t)$ (upper trace) and the scaled down radian frequency ω_2' (lower trace) for the case where the radian frequency offset $\Delta\omega$ is identical with the lock range $\Delta\omega_L$. An EXOR phase detector is used.

This signal is shown in the lower trace of Fig. 2.31. This signal modulates the frequency of the VCO such that its peak deviation becomes

$$\Delta\hat{\omega}_2 = F(\Delta\omega)K_0K_d\frac{\pi}{2} \tag{2.83}$$

When the division ratio of the (optional) down-scaler is N, the peak frequency deviation at the lower input of the phase detector (Fig. 2.1) gets $\Delta\hat{\omega}_2/N$. 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 $\Delta\omega$:

$$\Delta\hat{\omega}_2/N = \Delta\omega \tag{2.84}$$

Under this condition $\Delta\omega$ is identical with the lock range $\Delta\omega_L$. Combining Eqs. (2.83) and (2.84), we obtain the nonlinear equation

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

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

$$\Delta\omega_L \approx \pi\zeta\omega_n \tag{2.85}$$

for high-gain loops. Equation (2.85) is valid for any type of loop filter. For the lock time T_L we obtain the same approximation we obtained for the type 1 phase detector (Eq. 2.82).

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 $\Delta\omega_L$ for the EXOR is larger by a factor of $\pi/2$.

Phase detector type 3. The lock range of the PLL with a type 3 phase detector can be computed by an analysis similar to that for type 2 (EXOR). If we assume again that the PLL is initially out of lock and that the offset between reference frequency ω_1 and center frequency ω_0' is again $\Delta\omega$, the phase error becomes a ramp function again. Because the average output signal of the JK-flipflop $\overline{u_d}$ varies in a sawtooth-like fashion with phase error (Fig. 2.10), the average signal $\overline{u_d}$ will also be a sawtooth function, as shown in the upper curve of Fig. 2.32.

Now the frequency of the VCO is modulated in a sawtooth-like manner; see lower curve of Fig. 2.32. If the frequency offset $\Delta\omega$ is chosen such that the curve

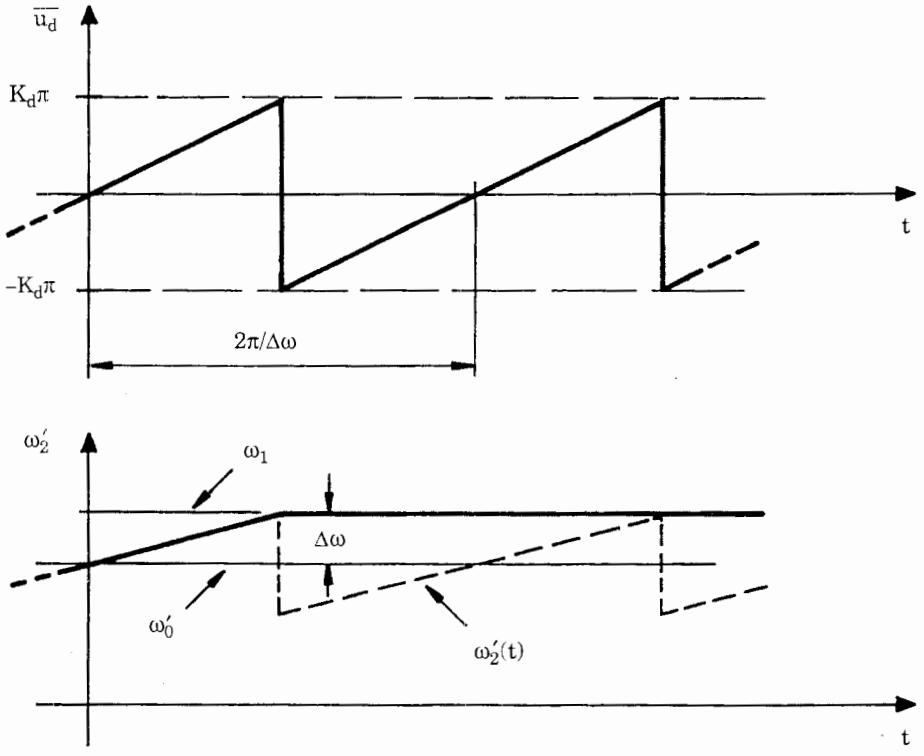


Figure 2.32 Waveforms of the average phase detector output signal $\overline{u_d}(t)$ (upper trace) and the scaled down radian frequency ω_2' (lower trace) for the case where the radian frequency offset $\Delta\omega$ is identical with the lock range $\Delta\omega_L$. A JK-flipflop phase detector is used.

just touches the ω_1 line, $\Delta\omega$ equals the lock range $\Delta\omega_L$. By analogy, we get the approximation

$$\Delta\omega_L \approx 2\pi\zeta\omega_n \tag{2.86}$$

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 Eq. (2.82) still holds true.

Phase detector type 4. The waveforms for $\overline{u_d}$ and ω_2' versus time for the PFD are shown in Fig. 2.33. Again the frequency of the VCO output signal is modulated in a sawtooth-like manner. In analogy to the preceding sections (EXOR and JK-flipflop phase detector) we get the following approximation for the lock range:

$$\Delta\omega_L \approx 4\pi\zeta\omega_n \tag{2.87}$$

For the lock-in time approximation, Eq. (2.82) is still valid.

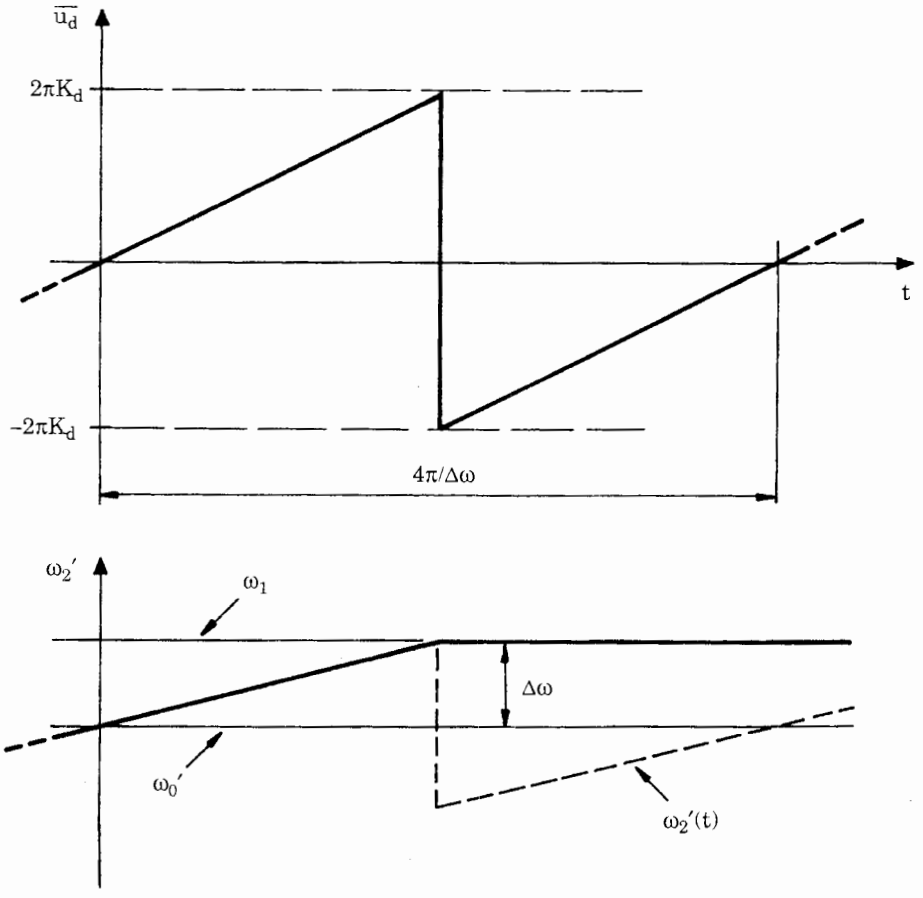


Figure 2.33 Waveforms of the average phase detector output signal $\overline{u_d}(t)$ (upper trace) and the scaled-down radian frequency ω_2' (lower trace) for the case where the radian frequency offset $\Delta\omega$ is identical with the lock range $\Delta\omega_L$. The PFD is used as phase detector.

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 ω_0. Consequently the average output signal $\overline{u_d}$ of the phase detector is a sine wave having the frequency $\Delta\omega$ and hence 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. 2.30.

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 ω_0. Under these conditions the PLL would never pull in!

Fortunately, we have overlooked the fact that the difference $\Delta\omega$ between reference frequency ω_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. 2.34. As a consequence, the average frequency of the VCO $\overline{\omega_2}$ is now higher than it was without any modulation; i.e., 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 ω_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 ω_2' finally reaches the reference frequency ω_1. This phenomenon is called the *pull-in process* (Fig. 2.35). 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 give only the final results. It is very important to note that the pull-in range depends on the type of loop filter.

Here are the formulas for the pull-in range:

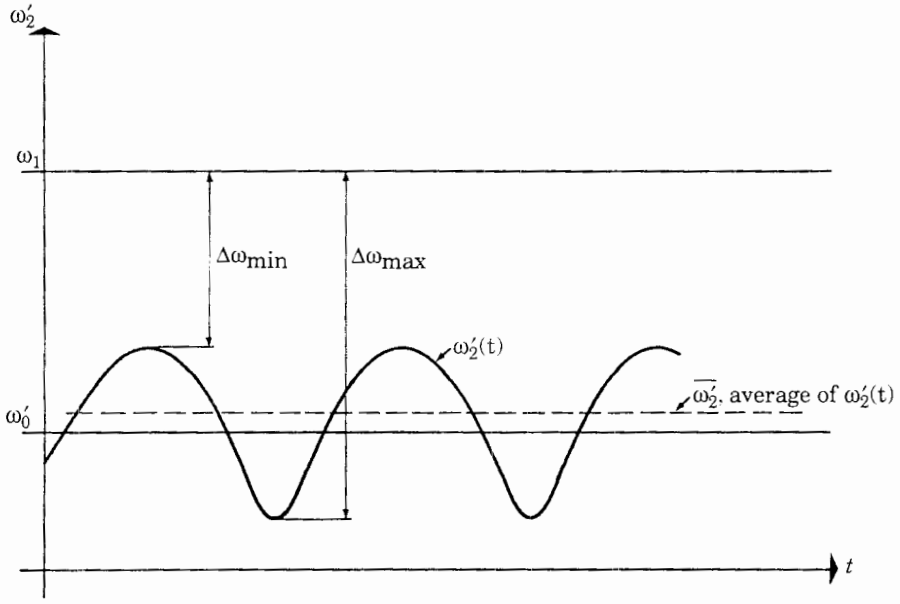


Figure 2.34 In the unlocked state of the PLL the frequency modulation of the VCO output signal is nonharmonic. This causes the average value of the VCO output frequency $\overline{\omega_2}$ to be pulled in the direction of the reference frequency.

- For the passive lead-lag filter

$$\Delta\omega_P \approx \frac{4}{\pi}\sqrt{2\zeta\omega_n K_0 K_d/N - \omega_n^2} \qquad \text{low-gain loops}$$
$$\Delta\omega_P \approx \frac{4\sqrt{2}}{\pi}\sqrt{\zeta\omega_n K_0 K_d/N} \qquad \text{high-gain loops} \tag{2.88}$$

- For the active lead-lag filter

$$\Delta\omega_P \approx \frac{4}{\pi}\sqrt{2\zeta\omega_n K_0 K_d/N - \omega_n^2/K_a} \qquad \text{low-gain loops}$$
$$\Delta\omega_P \approx \frac{4\sqrt{2}}{\pi}\sqrt{\zeta\omega_n K_0 K_d/N} \qquad \text{high-gain loops} \tag{2.89a}$$

- For the active PI filter

$$\Delta\omega_P \to \infty \tag{2.89b}$$

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.

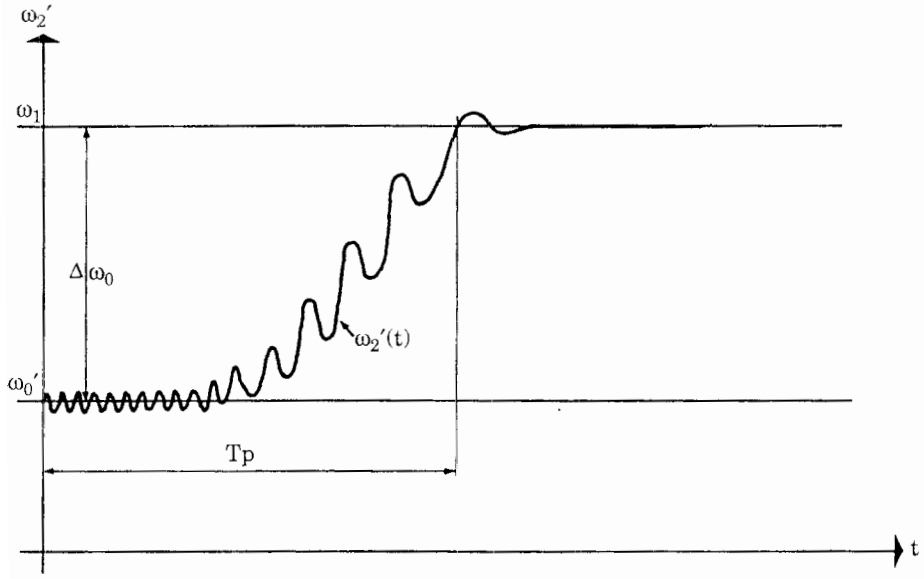


Figure 2.35 The pull-in process.

The pull-in process depicted in Fig. 2.35 can be easily explained by the mechanical analogy of Fig. 2.27. 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 characteristic of this process and is easily explained by the nonharmonic 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 valid only, 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

$$T_P \approx \frac{\pi^2}{16} \frac{\Delta\omega_0^2}{\zeta\omega_n^3} \tag{2.90}$$

- For the active lead-lag filter

$$T_P \approx \frac{\pi^2}{16} \frac{\Delta\omega_0^2 K_a}{\zeta \omega_n^3} \tag{2.91}$$

- For the active PI filter

$$T_P \approx \frac{\pi^2}{16} \frac{\Delta\omega_0^2}{\zeta \omega_n^3} \tag{2.92}$$

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. (2.90) to (2.92) 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; i.e., $N = 1$. Its natural frequency $f_n = \omega_n/2\pi$ is 3 Hz; this is a very narrowband system. The damping factor is chosen to be $\zeta = 0.7$. The loop gain K_0K_d/N is assumed to be $2\pi \cdot 1000$ rad/s. We shall now calculate the lock-in time T_L and the pull-in time T_P for an initial frequency offset Δf_0 of 30 Hz.

According to Eqs. (2.82) and (2.90), 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 procedure similar that done for the multiplier phase detector. We assume that the PLL is out of lock initially, that the VCO operates at its center frequency ω_0, and that the initial offset $\Delta\omega_0$ between reference frequency ω_1 and (down-scaled) VCO frequency ω_0' is large. The signals u_1 and u_2' can then be represented by

$$u_1(t) = U_{10}\ \text{rect}\ (\omega_0' t + \Delta\omega_0 t)$$
$$u_2'(t) = U_{20}\ \text{rect}\ (\omega_0' t)$$

respectively, where U_{10} and U_{20} are the amplitudes of the square wave signals. The phase error θ_e is the difference of the phases of these two signals; i.e.,

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

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. 2.36. (Let us ignore for the

moment the asymmetry of the waveform.) The output signal $u_f(t)$ 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. 2.36. If the triangular waveform were symmetrical (i.e., $T_1 = T_2$), the average frequency would remain constant and equal to ω_0'. As Fig. 2.36 demonstrates, however, the frequency offset $\Delta\omega(t)$ is not constant but is given by the difference between reference frequency ω_1 and *instantaneous* (scaled-down) VCO frequency ω_2'. Consequently, $\Delta\omega(t)$ becomes smaller during the positive half-wave of the $\overline{u_d}$ signal and larger during the negative half-wave. Therefore, the waveform becomes *asymmetrical,* which is exaggerated in Fig. 2.36. When the $\overline{u_d}$ waveform is asymmetrical, its mean value is no longer zero but becomes slightly positive. This causes the average frequency $\overline{\omega_2}$ of the VCO to be *pulled up*. Now, different things can happen. If the loop gain, i.e., the product $K_d K_0 F(0)/N$, is

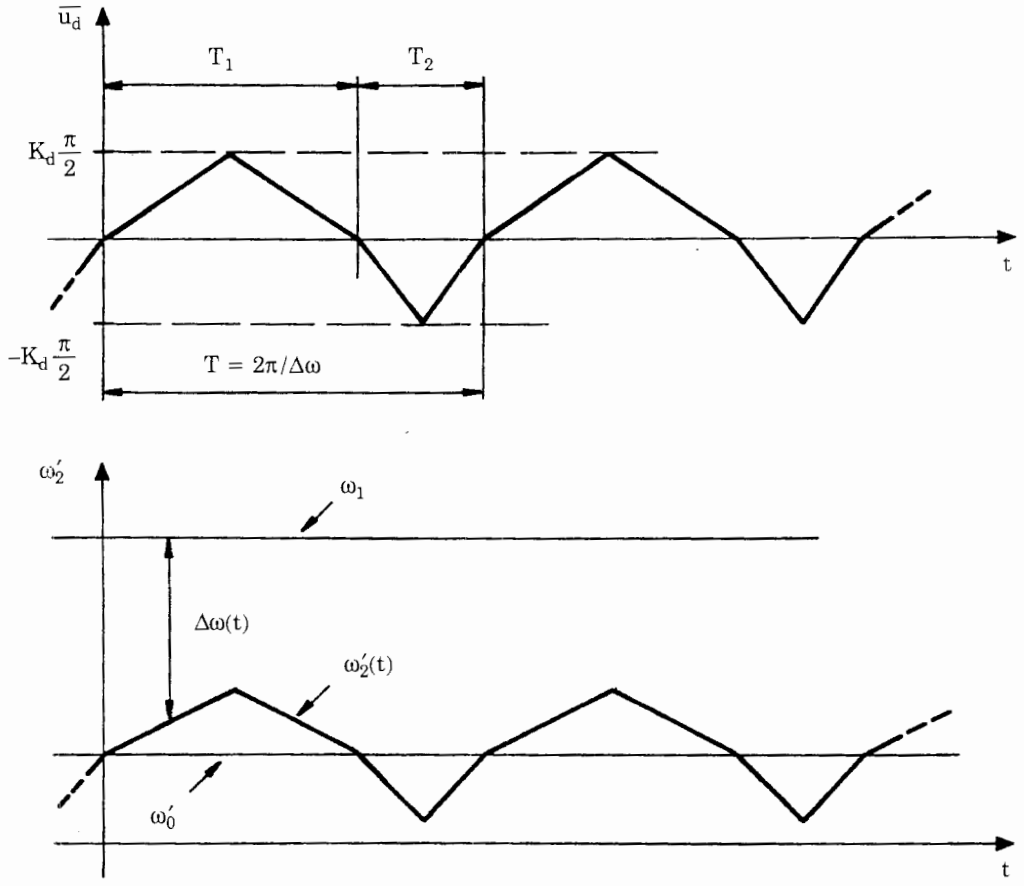


Figure 2.36 This figure is used to explain the slow pull-in process of the PLL, using the EXOR as phase detector. The upper trace shows the averaged output signal $\overline{u_d}$ of the EXOR gate, the lower trace the down-scaled instantaneous radian frequency $\omega_2'(t)$. The asymmetry of the waveforms is exaggerated. Because the duration of the positive half-wave of $\overline{u_d}$ 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; i.e., the peak value of ω_2' reaches ω_1 after some time.

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 $\overline{\omega_2}$ is pulled up so much that the $\overline{u_d}$ waveform becomes even more nonharmonic (i.e., the ratio of T_1/T_2 becomes significantly greater). This causes the mean $\overline{\omega_2}$ frequency to increase even more, and so forth, and the scaled-down VCO output frequency will be pulled up until it comes close to the reference frequency. Then 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 here (for details refer to App. A):

- Loop filter = passive lead-lag

$$\begin{aligned} \Delta\omega_P &\approx \frac{\pi}{2}\sqrt{2\zeta\omega_n K_0 K_d/N - \omega_n^2} && \text{low-gain loops} \\ \Delta\omega_P &\approx \frac{\pi}{\sqrt{2}}\sqrt{\zeta\omega_n K_0 K_d/N} && \text{high-gain loops} \end{aligned} \tag{2.93}$$

- Loop filter = active lead-lag

$$\begin{aligned} \Delta\omega_P &\approx \frac{\pi}{2}\sqrt{2\zeta\omega_n K_0 K_d/N - \omega_n^2/K_a} && \text{low-gain loops} \\ \Delta\omega_P &\approx \frac{\pi}{\sqrt{2}}\sqrt{\zeta\omega_n K_0 K_d/N} && \text{high-gain loops} \end{aligned} \tag{2.94}$$

- Loop filter = active PI

$$\Delta\omega_P \to \infty$$

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

$$\begin{aligned} T_P &\approx \frac{4}{\pi^2}\frac{\Delta\omega_0^2}{\zeta\omega_n^3} && \text{passive lead-lag filter} \\ T_P &\approx \frac{4}{\pi^2}\frac{\Delta\omega_0^2 K_a}{\zeta\omega_n^3} && \text{active lead-lag filter} \\ T_P &\approx \frac{4}{\pi^2}\frac{\Delta\omega_0^2}{\zeta\omega_n^3} && \text{active PI filter} \end{aligned} \tag{2.95}$$

As we know, the pull-in time becomes infinite when the initial frequency offset equals the pull-in range. When the passive or the active lead-lag filter is used, the approximation of Eq. (2.95) 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, "accept-

able" means that 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 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 ω_2' look like those drawn in Fig. 2.37.

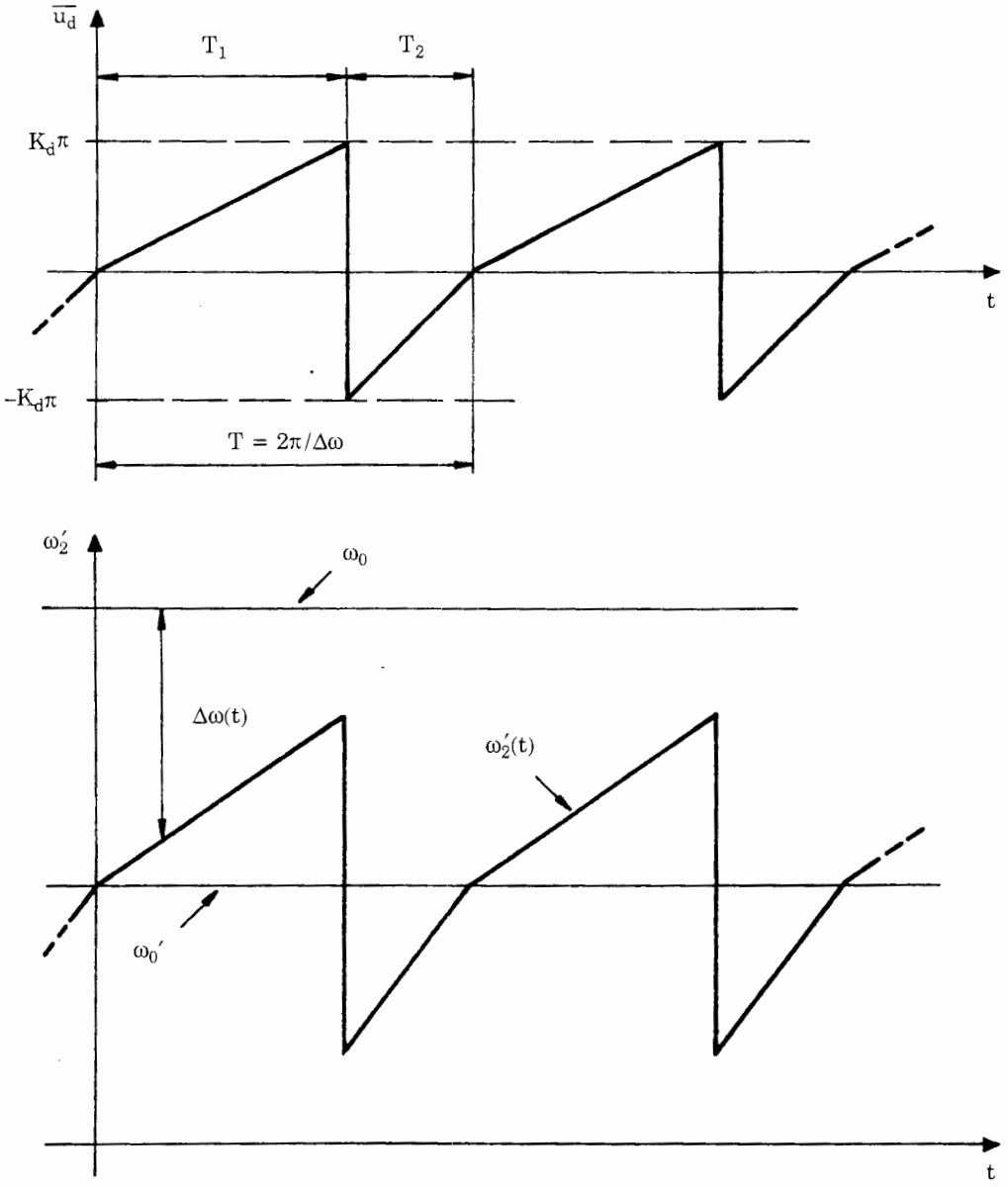


Figure 2.37 This figure explains the pull-in process of a PLL using the JK-flipflop as phase detector. The upper trace shows the average phase detector output signal $\overline{u_d}$. The lower trace shows the down-scaled frequency ω_2' created by the VCO.

Instead of triangular waves, we obtain sawtooth waves now. Performing an analog computation as above, we get for the pull-in range

- For the passive lead-lag filter

$$\begin{aligned} \Delta\omega_P &\approx \pi\sqrt{2\zeta\omega_n K_0 K_d/N - \omega_n^2} && \text{low-gain loops} \\ \Delta\omega_P &\approx \pi\sqrt{2}\sqrt{\zeta\omega_n K_0 K_d/N} && \text{high-gain loops} \end{aligned} \tag{2.96}$$

- For the active lead-lag filter

$$\begin{aligned} \Delta\omega_P &\approx \pi\sqrt{2\zeta\omega_n K_0 K_d/N - \omega_n^2/K_a} && \text{low-gain loops} \\ \Delta\omega_P &\approx \pi\sqrt{2}\sqrt{\zeta\omega_n K_0 K_d/N} && \text{high-gain loops} \end{aligned} \tag{2.97}$$

- For the active PI filter

$$\Delta\omega_P \to \infty \tag{2.98}$$

The pull-in time becomes

$$\begin{aligned} T_P &\approx \frac{1}{\pi^2}\frac{\Delta\omega_0^2}{\zeta\omega_n^3} && \text{passive lead-lag filter} \\ T_P &\approx \frac{1}{\pi^2}\frac{\Delta\omega_0^2 K_a}{\zeta\omega_n^3} && \text{active lead-lag filter} \\ T_P &\approx \frac{1}{\pi^2}\frac{\Delta\omega_0^2}{\zeta\omega_n^3} && \text{active PI filter} \end{aligned} \tag{2.99}$$

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

Phase detector type 4. When the PFD 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 (Fig. 2.15). As explained in Sec. 2.3.1 the PFD has a tristate output. When both flipflops (Fig. 2.11) are reset, the output is in the high-impedance state. Assume for the moment that a passive lead-lag loop filter is used in the PLL under consideration. When the PFD output floats, the charge on the capacitor remains unchanged; i.e., 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 tristate signal! In other words, the passive lead-lag filter has infinite gain at $f = 0$ under this condition. Volgers has shown[15] that the transfer function of the passive lead-lag filter must be

modified when it is driven by the PFD. It shows that the transfer function is approximately given by

$$F(s) \approx \frac{1 + s\tau_2}{s(\tau_1 + \tau_2)} \tag{2.100}$$

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

$$F(s) \approx -K_a \frac{1 + s\tau_2}{s\tau_1} \tag{2.101}$$

where $K_a = C_1/C_2$ (Fig. 2.16*b*). 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 tristate output. In any case, the PI filter acts as an integrator whose transfer function is given by

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

Because all loop filters behave as real integrators now, the pull-in range $\Delta\omega_P$ becomes theoretically *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 are going to 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 ω_0. The input radian frequency ω_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. 2.38. As explained in Sec. 2.3.1 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 that the logical high level is U_B and 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.

Obviously, the average duty cycle of u_d is 50 percent. As Fig. 2.15 shows, the average duty cycle δ varies very little with the ratio ω_1/ω_2' and can be considered constant for this analysis. Because the time constant τ_1 of the loop filter is much larger than the period of the u_1 signal in Fig. 2.38, 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. 2.38. If the equivalent signal u_{eq} had a duty cycle of 100 percent, capacitor C of the loop filter would simply charge toward the supply voltage U_B with time constant $\tau_1 + \tau_2 =$

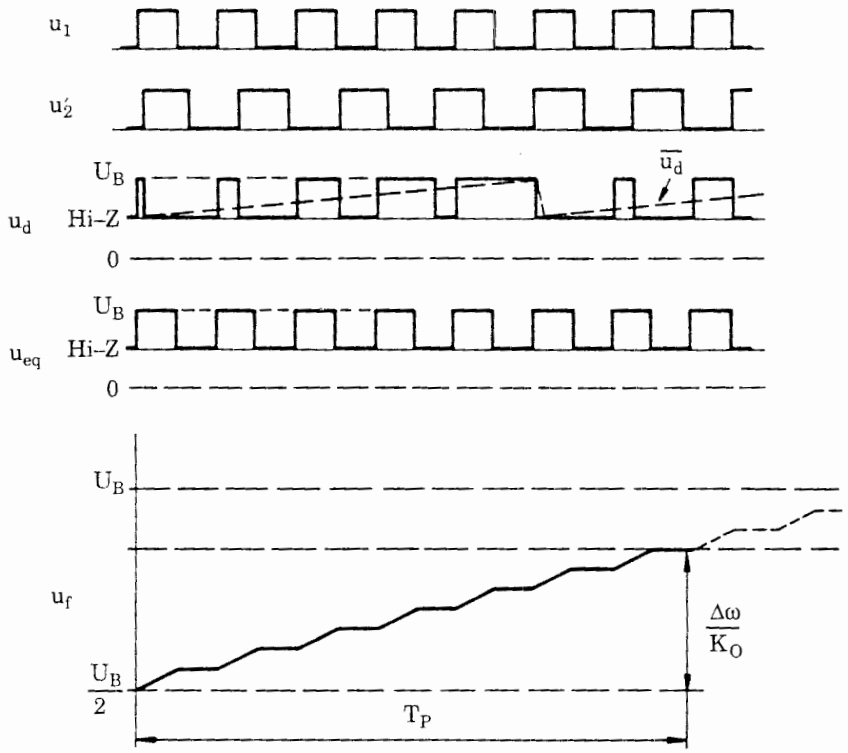


Figure 2.38 The waveforms shown in this figure demonstrate the pull-in process of a PLL that uses the PFD. The frequency ω_1 of the input signal u_1 is assumed to be higher than the (scaled-down) frequency ω_2' of the VCO output signal. Consequently, u_d toggles between the states 0 and 1 (third trace). The average u_d signal has the same effect as an equivalent signal u_{eq} with 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 bottom trace.

$(R_1 + R_2)C$, because C is charged through the series connection of resistors R_1 and R_2 (compare Fig. 2.16*a* 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. 2.39 can now be used to compute the signal u_f across capacitor C. Since the VCO of the PLL was assumed to operate at its center frequency ω_0 at the start of the pull-in process, the initial 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. 2.39 is therefore $U_B/2$.

To get locked, the VCO must create an output frequency that is offset from the center frequency ω_0 by the amount $N\,\Delta\omega_0$. To obtain that frequency, the voltage on capacitor C must be increased by $N\,\Delta\omega_0/K_0$ [compare with Eq. (2.32)]. The pull-in time T_P now is simply the time after which the voltage on capacitor C has reached that level.

For the passive lead-lag filter the calculation yields

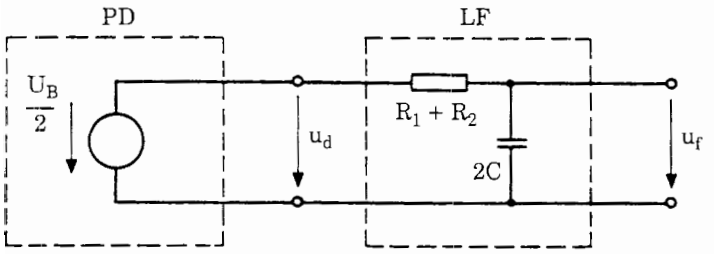


Figure 2.39 Charging of capacitor C in the loop filter is calculated by this equivalent model. Detailed explanations are in the text.

- Loop filter = passive lead-lag

$$T_P \approx 2(\tau_1 + \tau_2) \ln \frac{1}{1 - \dfrac{2N\,\Delta\omega_0}{U_B K_0}} \tag{2.102}$$

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_{sat+} (positive) and U_{sat-} (negative), respectively. To account for clipping, we simply have to replace $U_B/2$ in Eq. (2.102) by $(U_{sat+} - U_{sat-})/2$.

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

- Loop filter = active lead-lag

$$T_P \approx 2\tau_1 \ln \frac{1}{1 - \dfrac{2N\,\Delta\omega_0}{U_B K_0 K_a}} \tag{2.103}$$

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

- Loop filter = active PI

$$T_P \approx \frac{4\tau_1 \Delta\omega_0 N}{K_0 U_B} \tag{2.104}$$

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. 2.36. 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. 2.35 for the multiplier phase detector. A similar pumping is observed when the phase detector is a JK-flipflop (Fig. 2.37). No pumping occurs, however, when the PFD is used. Since the driving signal for the loop filter is unipolar here (Fig. 2.39), 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 (Chap. 5).

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, i.e., a filter having a pole at $s = 0$ (or in other words, a filter having "infinite gain" at dc). As we already have seen, the pull-in range cannot be larger than the range of frequencies the VCO is able to create. But 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. Let us explain that phenomenon by two examples.

Example 1 Consider a PLL with type 1 phase detector (multiplier) and no down-scaler. Let the center frequency be $f_0 = 100\,\text{kHz}$. Assume that the reference frequency is $f_1 = 100\,\text{kHz}$ 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 $f_1 = 100\,\text{kHz}$ and $f_2 = 50\,\text{kHz}$ the multiplier generates the sum and difference of these two frequencies, i.e., 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 and 150 kHz. The lower sideband (50 kHz) has exactly the reference frequency, which causes the phase detector to output a dc signal that 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 $f_0 = 100\,\text{kHz}$. 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 now? 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, i.e., third, fifth, etc. The third harmonic is 120 kHz. When the VCO oscillates at 120 kHz, the EXOR phase detector thinks that the PLL is perfectly locked. Indeed it *is* locked, but on the *wrong* frequency.

It turns out that the full pull-in range can be realized only 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 that frequency step that causes a lockout if applied to the reference input of the PLL. In the mechanical analogy of Fig. 2.27, the pull-out frequency corresponds to that 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[6] have led to an approximation:

$$\Delta\omega_{PO} \approx 1.8\omega_n(\zeta + 1) \tag{2.105}$$

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

$$\Delta\omega_L < \Delta\omega_{PO} < \Delta\omega_P \tag{2.106}$$

If, in an FM system, the PLL is pulled out by too large a frequency step, we can expect that PLL to come back 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. 2.27 to tip over. With the EXOR phase detector the average output signal $\overline{u_d}$ 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 explicitely the pull-out frequency. With 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

$$\Delta\omega_{PO} \approx 2.46\omega_n(\zeta + 0.65) \tag{2.107}$$

As could be expected, this result is not far away 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 π. Because the average output signal $\overline{u_d}$ of the JK-flipflop actually is *linear* in the range $-\pi < \theta_e < \pi$, the pull-out range can be computed explicitly. By using the linear model of the PLL (Fig. 2.21), phase error θ_e is calculated for a frequency step $\Delta\omega$ applied to the reference input. The result is a damped oscillation.[1] By using the rules of differential calculus, it is straight-

forward to calculate the maximum of the phase error. From there it is quite easy to calculate the size of frequency step that leads to a peak phase error of π. Assuming that the PLL is a high-gain loop, we get

$$\Delta\omega_{PO} = \begin{cases} \pi\omega_n \exp\left(\dfrac{\zeta}{\sqrt{1-\zeta^2}} \tan^{-1} \dfrac{1-\zeta^2}{\zeta}\right) & \zeta < 1 \\ \pi\omega_n e & \zeta = 1 \\ \pi\omega_n \exp\left(\dfrac{\zeta}{\sqrt{\zeta^2-1}} \tanh^{-1} \dfrac{\zeta^2-1}{\zeta}\right) & \zeta > 1 \end{cases} \tag{2.108}$$

If $\Delta\omega_{PO}$ is plotted against ζ, 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. (2.108) gave the approximation

$$\Delta\omega_{PO} \approx 5.78\omega_n(\zeta + 0.5) \tag{2.109}$$

Phase detector type 4. In the case of the PFD, the pull-out range is the frequency step causing the peak phase error to exceed 2π. Because the average output signal $\overline{u_d}$ 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

$$\Delta\omega_{PO} = \begin{cases} 2\pi\omega_n \exp\left(\dfrac{\zeta}{\sqrt{1-\zeta^2}} \tan^{-1} \dfrac{1-\zeta^2}{\zeta}\right) & \zeta < 1 \\ 2\pi\omega_n e & \zeta = 1 \\ 2\pi\omega_n \exp\left(\dfrac{\zeta}{\sqrt{\zeta^2-1}} \tanh^{-1} \dfrac{\zeta^2-1}{\zeta}\right) & \zeta > 1 \end{cases} \tag{2.110}$$

Here, the least-squares fit gave the linear approximation

$$\Delta\omega_{PO} \approx 11.55\omega_n(\zeta + 0.5) \tag{2.111}$$

To ease the design of mixed-signal PLLs, the equations derived in Sec. 2.6 are summarized in four tables, one table for each type of phase detector. Table 2.1 lists the formulas for the most important key parameters for PLLs using phase detector type 1, Table 2.2 contains the corresponding equations for the Exor phase detector, Table 2.3 for phase detector 3, and Table 2.4 for phase detector 4.

2.7 Phase Detectors with Charge Pump Output

When analyzing the pull-in range of the PLL (Sec. 2.6.1), we found that a wide pull-in range is most easily achieved by using the PFD as phase detector.

TABLE 2.1 Summary of Parameters of the Mixed-Signal PLL, Phase Detector Type 1 (Multiplier)

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

Moreover, it showed that the passive lead-lag loop filter performs like a real integrator when driven by the PFD [Eq. (2.100)]. This is because the charge on the loop filter capacitor remains unchanged when the output of the PFD is in the high-impedance state (if we discard leakage currents for the moment). Therefore in most digital PLLs the combination of PFD and passive lead-lag filter is the preferred arrangement (Fig. 2.40*a*).

As reported by Volgers[15] this circuit has a property that can be disturbing in critical applications: the phase detector gain K_d of the PFD is not constant as predicted by theory, but varies with the "operating point" of the loop. By "operating point" we mean the average loop filter output signal $\overline{u_f}$ that is required to create the desired output frequency ω_2 of the VCO.

Let us explain that by a simple consideration. Assume that the PLL is powered from a unipolar supply with $U_B = 5\,\text{V}$, and that the PLL initially operates at its center frequency ω_0. Under this condition, the output signal u_f of the loop filter will settle at half the supply voltage, i.e., at $u_f = 2.5\,\text{V}$. Capacitor C therefore gets charged to 2.5 V. When the PLL has become locked, the output of the PFD will be in the high-impedance state most of the time. The PFD will

TABLE 2.2 Summary of Parameters of the Mixed-Signal PLL, Phase Detector Type 2 (EXOR)

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

occasionally switch into the high or low state during very short intervals of time just to keep the voltage on capacitor C at the proper level. Suppose now that the PLL is required to pull up the output frequency ω_2 to a much higher level. The PFD will therefore switch to the high state; i.e., its output voltage will become $u_d \approx 5\,\text{V}$ for some time. Capacitor C will then be charged to a higher voltage through the series connection of resistors R_1 and R_2. If we assume that the sum $R_1 + R_2$ equals $2.5\,\text{k}\Omega$, a current of $2.5\,\text{V}/2.5\,\text{k}\Omega = 1\,\text{mA}$ will initially flow into capacitor C. The capacitor will charge toward U_B with a time constant $(R_1 + R_2)C$.

Let us suppose now that the PLL did not operate at its center frequency initially, but that its output frequency was offset so much that the average loop filter output signal u_f had to be pulled up to, say, 4.0 V to maintain the desired output frequency ω_2. If the PLL is required now to pull up its output frequency even more, the PFD will switch again into the high state ($u_d \sim 5\,\text{V}$). Again, capacitor C will charge with time constant $(R_1 + R_2)C$ toward 5 V, but the current that initially flows into capacitor C now is much smaller now than in the former example. Because there is a voltage difference of only $5 - 4 = 1\,\text{V}$ across

TABLE 2.3 Summary of Parameters of the Mixed-Signal PLL, Phase Detector Type 3 (JK-Flipflop)

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

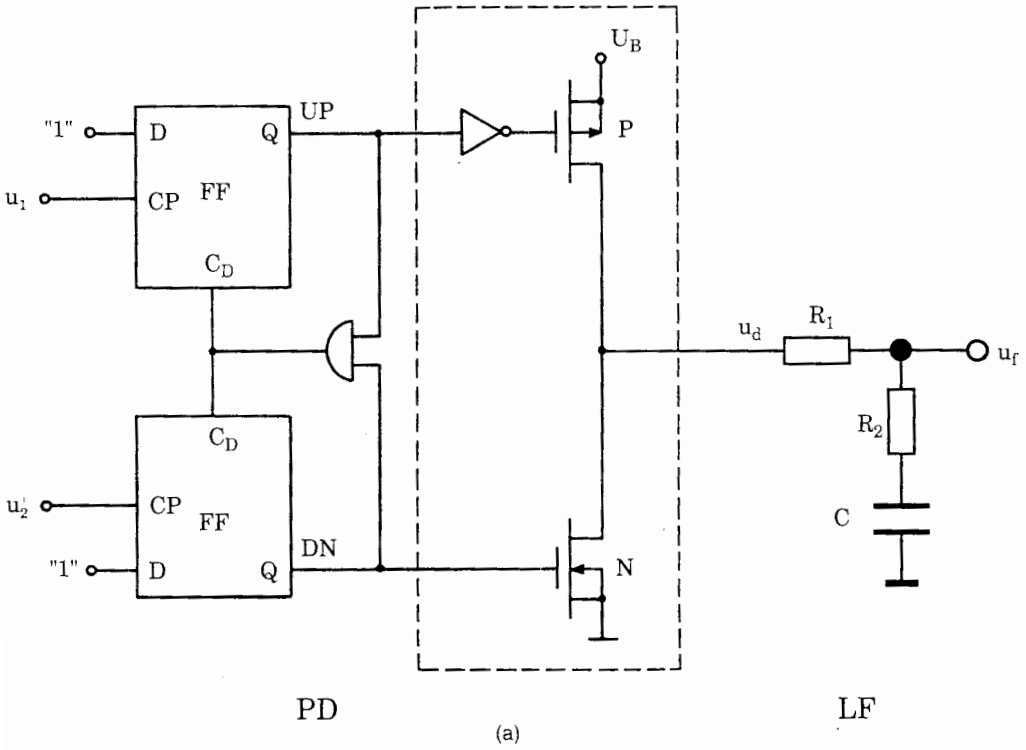
resistors $R_1 + R_2$, the current flowing into C is only 1 V/2.5 kΩ = 0.4 mA now. This implies that the loop gain of the PLL has become smaller by a factor 2.5 in this situation.

When we consider the opposite case where the PLL initially operated at a frequency much below the center frequency, the average u_f signal would have been set to perhaps 1.0 V. When the PLL is required to increase its output frequency, the initial current flowing into capacitor C would be increased to 4 V/2.5 kΩ = 1.6 mA. Here the loop gain is higher by a factor 1.6 when compared with the first case. This simple example demonstrates that the phase detector gain K_d of the PFD is not a constant, but varies with the initial output signal $\overline{u_f}$. When the phase detector gain K_d is reduced, the loop gain (which is given by $K_d K_0/N$) is reduced as well. A lower loop gain leads to a smaller natural frequency ω_n and a smaller damping factor ζ, which could produce unacceptable overshoot.

There are several ways to overcome the problem of varying phase detector gain. One option would be to use the active PI filter instead of the passive lead-lag, as shown in Figure 2.40*b*. In this circuit the output current delivered by

TABLE 2.4 Summary of Parameters of the Mixed-Signal PLL, Phase Detector Type 4 (PFD)

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



(a)

Figure 2.40*a* Most digital PLLs use the PFD combined with the passive lead-lag loop filter.

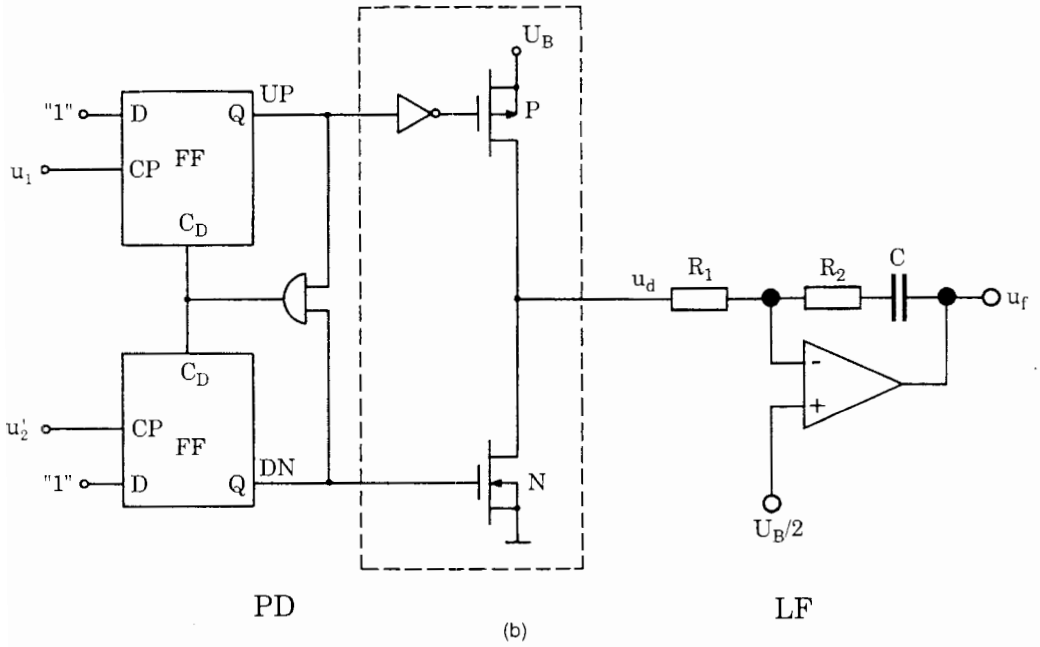
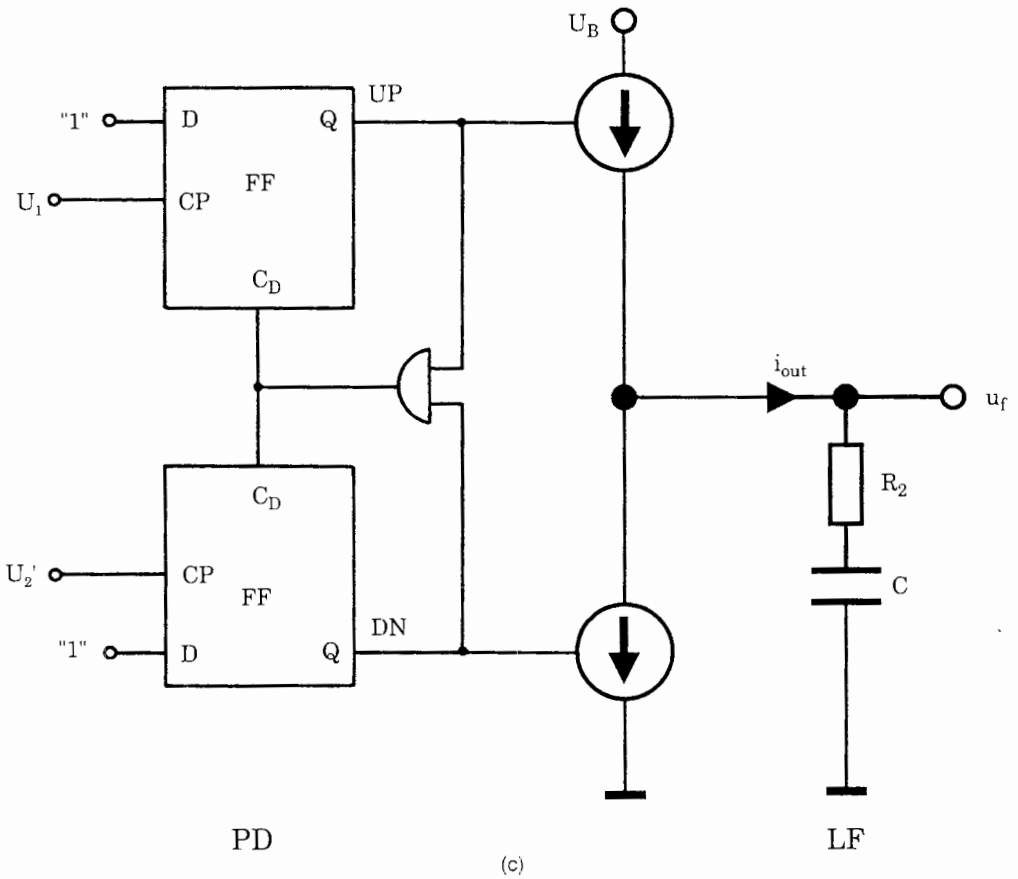


Figure 2.40*b* Combination of the PFD with the active PI loop filter.

the PFD always flows into the summing junction of the operational amplifier, which is at a constant level of $U_B/2 = 2.5\,\text{V}$; hence that current does not depend on the output level u_f.

There is another way to go around that problem: the application of a phase detector having a current output instead of voltage output. Such phase detectors are said to have a *charge pump* output. Figure 2.40*c* shows the cascade of a charge pump PFD with a passive lead-lag loop filter. The PFD shown on the left side is built up like a conventional PFD, but the outputs of the UP and DN flipflops are controlling two current sources. When the UP flipflop is set, the upper current source (right top in Fig. 2.40*c*) sources current into the output terminal. When the DN flipflop is set, however, the lower current source sinks current from the output terminal. The PFD output current $\overline{i_{\text{out}}}$ averaged over one reference cycle is given by $\overline{u_d}/R_b$, where $\overline{u_d}$ is the average voltage output signal that would be delivered by a conventional PFD (i.e., by a PFD having voltage output), and R_b is the transimpedance of the voltage-to-current converter inherent in this type of PFD. The transimpedance R_b is usually set by an external resistor, as explained, e.g., in Ref. 51.

The average current output versus phase error is given by $\overline{i_{\text{out}}} = K_d\theta_e/R_b$. To get the average loop filter output signal $\overline{u_f}$, we must multiply the PFD output current by the impedance of the loop filter, which is simply the sum of resistor R_2 and the reactance $1/sC$ of capacitor C. For the loop filter output signal we get



(c)

Figure 2.40c PFD with charge pump output, followed by passive lead-lag loop filter.

$$U_f(s) = \Theta_e(s) K_d \frac{1 + sR_2C}{sR_bC}$$

Putting $\tau_1 = R_bC$ and $\tau_2 = R_2C$, we get the simple expression

$$U_f(s) = \Theta_e(s) K_d \frac{1 + s\tau_2}{s\tau_1}$$

This result is identical with that obtained for a digital PLL built from a voltage output PFD followed by an active PI loop filter. Consequently we can use the formulas given in Table 2.4 for the active PI loop filter to compute the relevant key parameters of the PLL such as lock range, pull-in range, and the like.

There are a number of commercially available PLLs utilizing a PFD having a charge pump output, e.g., the 74HCT9046A manufactured by Philips[51] (see also Table 10.1). Phase detectors with charge pump output can be combined with any kind of loop filters, passive or active. Passive filters are preferred in most cases; because the current output signal of the charge pump is a tristate

signal, a passive RD filter connected to the charge pump output behaves like a real integrator. Hence an active filter would not bring additional benefit, at least not for first-order filters. In Sec. 4.7 we will discuss higher-order loop filters that can be cascaded with charge pump outputs.

2.8 PLL Performance in the Presence of Noise

One of the most intriguing properties of the PLL is its ability to extract signals from an extremely noisy environment. This property is made use of extensively in the whole field of communications. The theory of noise in PLLs is extremely cumbersome. Exact solutions for noise performance have been derived for first-order PLLs only[4]; for second-order PLLs, computer simulations have been made, which have provided approximate results.[5] We deal here only with the simplified noise theory, as presented, e.g., by Gardner and Viterbi.[1,4] Whereas Gardner and Viterbi analyzed noise performance of linear PLLs (i.e., PLLs using a type 1 phase detector), Selle and coworkers investigated noise in digital PLLs in more detail.[24] Before entering into details, we first want to identify the sources and types of noise in PLL systems.

2.8.1 Sources and types of noise in a PLL

As was demonstrated by Rohde,[48] every building block of the PLL can contribute to noise. In applications of communication, however, the noise superimposed to the input signal (reference signal) is predominant, so we will restrict ourselves to reference noise. When the PLL is used as a frequency synthesizer (see Chap. 3), the reference source is normally a high-quality oscillator whose frequency is usually determined by a quartz crystal. As shown by Rohde even the highest-quality oscillators can create noise, due to thermal noise of transistors and resistors within the oscillator circuit.[48] Also, here the noise source is the reference signal itself.

In most communications systems in which the PLL is made use of, digital signals are transmitted; i.e., the baseband signals consist of pulses or square wave signals. A binary sequence of 10101010 . . . , e.g., would be represented as a symmetrical square wave. Because of the harmonics, the bandwidth requirements would become excessive it the pure square wave signal were sent. To save bandwidth, only the fundamental is transmitted. The received signal therefore is a sine wave in this case. 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 considerer "analog."

In every communication link, the transmitted signal picks up noise. Noise can be generated in repeaters, but it also can be created by crosstalk from other channels, from atmospheric noise, and from many other sources. Anyway, all that noise is added algebraically to the (analog) data signal. This is sketched by Fig. 2.41 for the trivial case that the data signal is a bit stream of the form 10101010. . . . There are many types of additive noise. The most common is

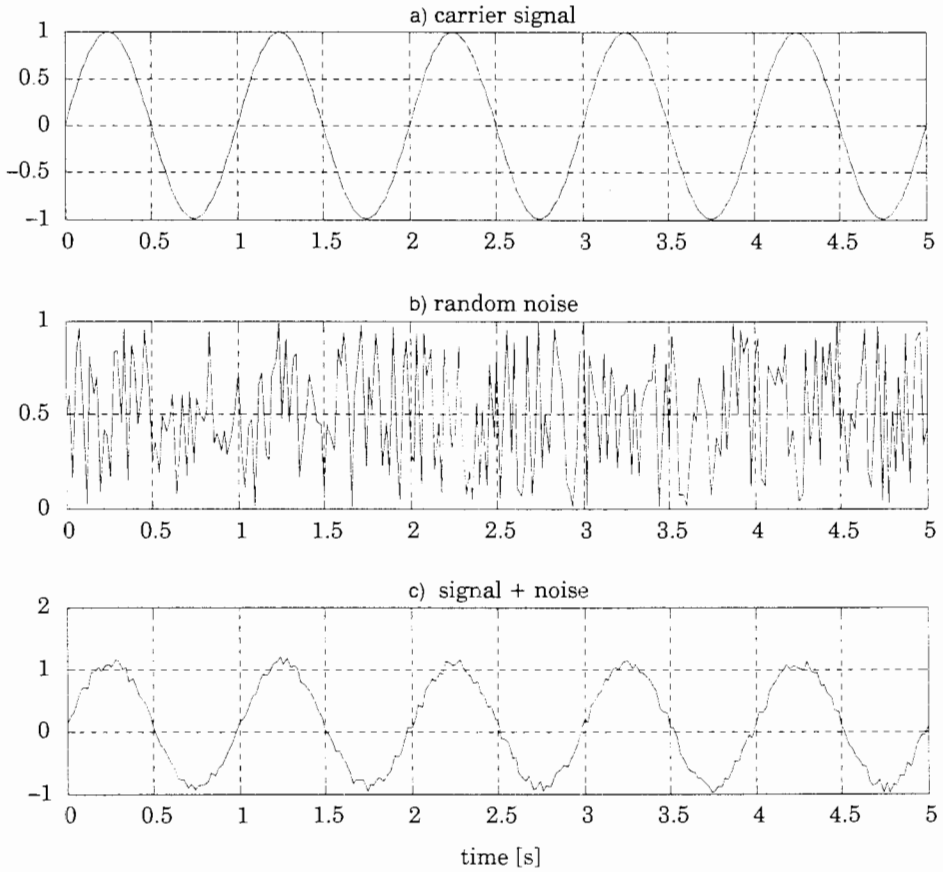


Figure 2.41 Added noise to information signal. (*a*) A very simple signal (a sequence of binary 101010 . . .). (*b*) A random noise signal. (*c*) The sum of both.

called *added white gaussian noise* (AWGN). The term *gaussian* characterizes the amplitude distribution of the signal, indicating 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 indicates that within the so-called noise bandwidth of the channel, every frequency interval df contains the same noise power dP_n; i.e., dP_n/df is constant. The noise shown in Fig. 2.41 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 by using, for example, a Schmitt trigger. The converted signal is then a square wave. For a binary signal, its amplitude can take only two levels, "high" and "low." The noise superimposed on the data signal now causes the zero crossings of the square wave to become "jittered," as shown in Fig. 2.42.

This kind of noise is called *phase noise* or *phase jitter*. The solid curve in Fig. 2.42 is the undistorted data signal, and the thinner lines represent the jittered transients. To cope with the effects of superimposed noise, we now shall define the most important parameters that describe noise.

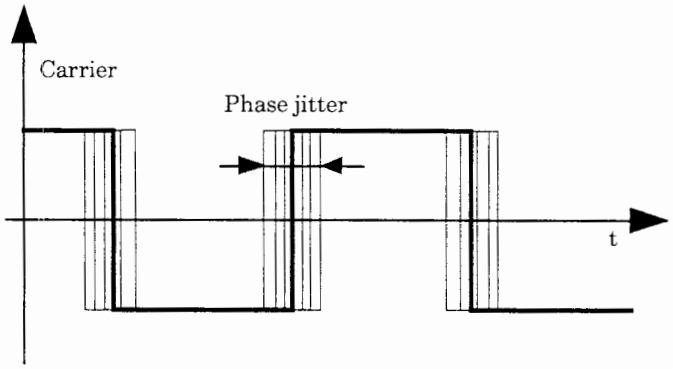


Figure 2.42 When a data signal is reshaped, amplitude noise is converted to phase noise (phase jitter).

2.8.2 Defining Noise Parameters

To analyze noise performance of the PLL, it is most convenient to describe signals and noise by their power density spectra, Fig. 2.43. 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 $B_s/2$. The noise spectrum usually is also symmetrical around the center frequency and is broadband in most cases; its one-sided bandwidth is labelled $B_i/2$. Signal power is denoted P_s in this figure; noise power is denoted 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 $\theta_1(t)$. For $s = j\omega$, $H(\omega)$ is the *phase frequency response* of the PLL. As we see from the plot in Fig. 2.22, $H(\omega)$ is a lowpass filter function. Now the variable ω in $H(\omega)$ is the (radian) frequency of the phase signal that modulates the carrier frequency ω_0'; hence ω 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 Fig. 2.43. (Note that H is plotted versus frequency f here and not versus radian frequency ω.) 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. 2.8.4. 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. 2.43, 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 section, there are two important parameters describing 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

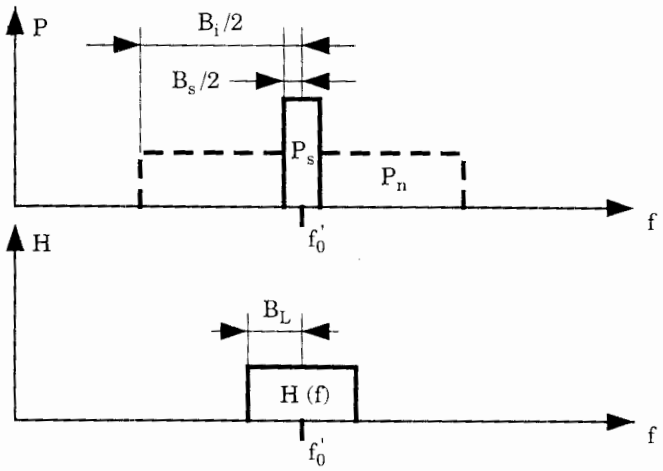


Figure 2.43 Definition of noise parameters; for meaning of symbols see text.

2.8.3 The impact of noise on PLL performance

The noise analysis presented here is based on the work of Gardner.[1] We are going to 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 a single spectral line at a frequency f_0, the one-sided power density spectrum shows 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 two-sided power spectra are used, 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, 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. 2.44, where input signal and input noise are plotted once again. Here we assume for simplicity that the input signal consists of one spectral line at the down-scaled center frequency f_0'. 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 (see input u_1 in Fig. 2.1). Assuming that 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[1]

$$\overline{\theta_{n1}^2} = \frac{P_n}{2P_s} \tag{2.112}$$

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.

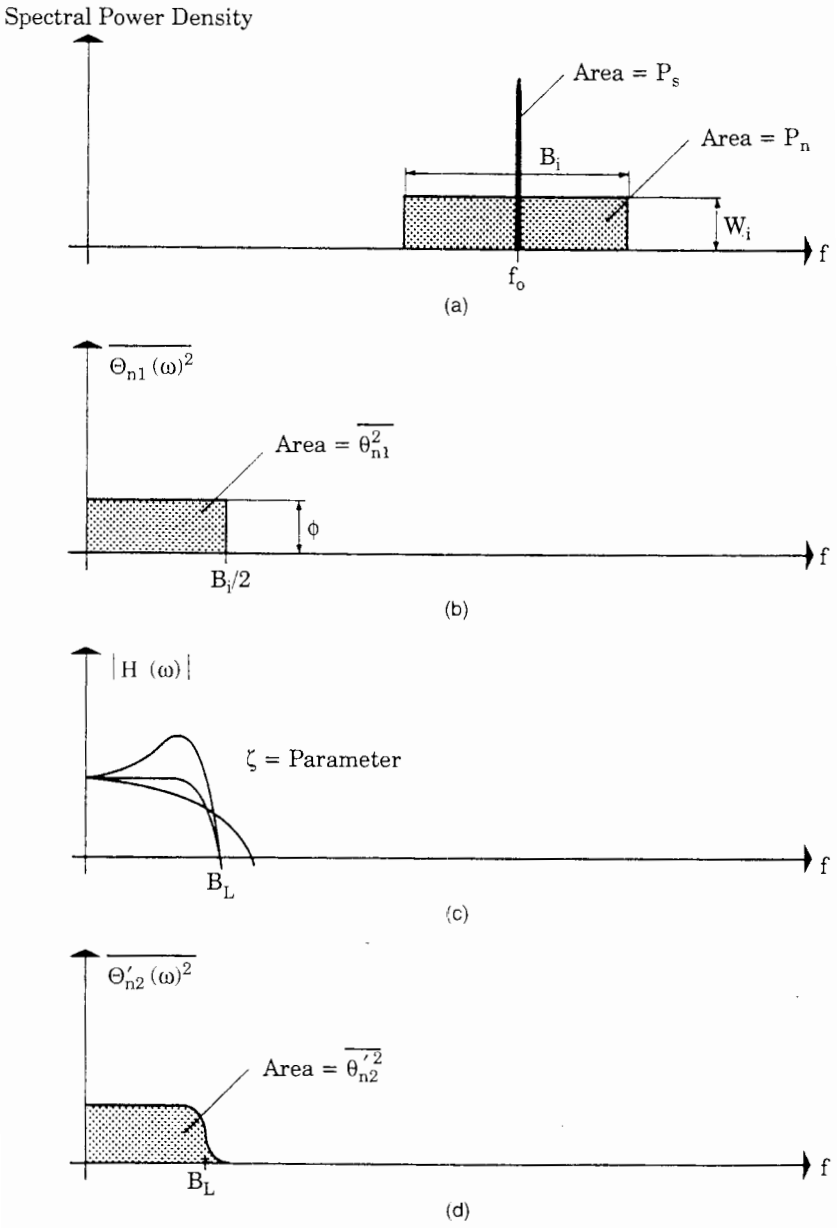


Figure 2.44 Method of calculating the phase jitter $\overline{\theta_{n2}'^2}$ at the down-scaled output of the PLL. (*a*) Power spectra of the reference signal $u_1(t)$ and the superimposed noise signal $u_n(t)$. (*b*) Spectrum of the phase noise at the input of the PLL. (*c*) Bode plot of the phase-transfer function $H(\omega)$. (*d*) Spectrum of the phase noise at the down-scaled output of the PLL.

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

$$(\text{SNR})_i = \frac{P_s}{P_n} \tag{2.113}$$

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

$$\overline{\theta_{n1}^2} = \frac{1}{2(\text{SNR})_i} \quad \text{rad}^2 \tag{2.114}$$

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'.[1] Fig. 2.44*b* shows the mean square of input phase jitter:

$$\overline{\Theta_{n1}^2(\omega)} = \Phi = \frac{\overline{\theta_{n1}^2}}{B_i/2} \quad \text{rad}^2 \cdot \text{Hz}^{-1} \tag{2.115}$$

Since the noise spectrum has been assumed white, the mean square $\overline{\Theta_{n1}^2(\omega)}$ has the constant value Φ. 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

$$\overline{\Theta_{n1}^2{}'(\omega)} = |H(\omega)|^2 \cdot \overline{\Theta_{n1}^2(\omega)} = |H(\omega)|^2 \Phi \tag{2.116}$$

Fig. 2.44*c* plots the Bode diagram of $H(\omega)$, and Fig. 2.44*d* 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 output phase jitter $\overline{\theta_{n2}^2{}'}$, we use the Parseval theorem to get

$$\overline{\theta_{n2}^2{}'} = \int_0^\infty \overline{\Theta_{n2}^2{}'}(2\pi f)\, df \tag{2.117}$$

where $2\pi f = \omega$. $\overline{\theta_{n2}^2{}'}$ is the area under the curve in Fig. 2.44*d*. Making use of Eqs. (2.116) and (2.117), we get

$$\overline{\theta_{n2}^2{}'} = \int_0^\infty \Phi |H(2\pi f)|^2 df = \frac{\Phi}{2\pi} \int_0^\infty |H(\omega)|^2 d\omega \tag{2.118}$$

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

$$B_L = \int_0^\infty |H(2\pi f)|^2 df \tag{2.119}$$

The solution of this integral reads[1]

$$B_L = \frac{\omega_n}{2}\left(\zeta + \frac{1}{4\zeta}\right) \tag{2.120}$$

Thus B_L is proportional to the natural frequency ω_n of the PLL; furthermore, it depends on the damping factor ζ. Figure 2.45 is a plot of B_L versus ζ, and B_L has a minimum at $\zeta = 0.5$. In this case we have

$$B_L = B_{L\min} = \frac{\omega_n}{2} \tag{2.121}$$

In Sec. 2.3 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 $\overline{\theta_{n2}^2}'$ we can now write

$$\overline{\theta_{n2}^2}' = \Phi B_L \tag{2.122}$$

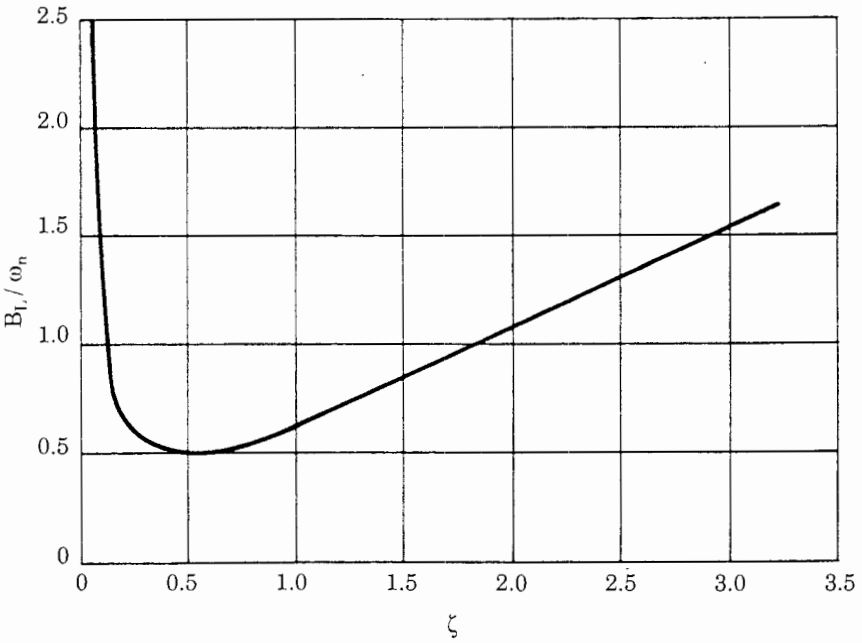


Figure 2.45 Noise bandwidth of a second-order PLL as a function of the damping factor ζ. (*Adapted from Gardner*[1] *with permission.*)

where Φ is already known from Eq. (2.115). Combining Eqs. (2.112), (2.115), and (2.122), we obtain

$$\overline{\theta_{n2}^{2}}' = \frac{P_n}{P_s} \cdot \frac{B_L}{B_i} \tag{2.123}$$

We saw in Eq. (2.114) that the phase jitter at the input of the PLL is inversely proportional to the $(\text{SNR})_i$. By analogy we can also define a signal-to-noise ratio at the output, which will be denoted by $(\text{SNR})_L$ (SNR of the loop). We define this analogy in Eq. (2.124),

$$\overline{\theta_{n2}^{2}}' = \frac{1}{2(\text{SNR})_L} \tag{2.124}$$

Comparing Eqs. (2.123) and (2.124), we get

$$(\text{SNR})_L = \frac{P_s}{P_n} \cdot \frac{B_L}{B_i} = (\text{SNR})_L \cdot \frac{B_i}{2B_L} \tag{2.125}$$

Equation (2.125) 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[1]:

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

In quantitative terms, according to Eq. (2.124), for $(\text{SNR})_L = 4$ the output phase noise becomes

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

Hence the rms value $\sqrt{\overline{\theta_{n2}^{2}}'}$ becomes

$$\sqrt{\overline{\theta_{n2}^{2}}'} = \sqrt{1/8} = 0.353 \quad \text{rad}$$

Since the rms value of phase jitter is about 20°, the value of 180° (limit of dynamic stability) is rarely exceeded. Consequently the PLL does not unlock

frequently. At $(SNR)_L = 1$, however, the effective value of output phase jitter would be as large as 40°, and the dynamic limit of stability of would be exceeded on every major noise peak, thus making stable operation impossible.

As a rule of thumb,

$$(SNR)_L \geq 4\,(6\,dB) \tag{2.126}$$

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

$$SNR = \frac{P_s}{P_n} = \frac{U_s^2\,(rms)}{U_s^2\,(rms)}$$

whereas the $(SNR)_{dB}$ is calculated from

$$(SNR)_{dB} = 10 \log_{10} \frac{P_s}{P_n} = 20 \log_{10} \frac{U_s\,(rms)}{U_n\,(rms)}$$

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 lockouts. For example, if $T_{av} = 100\,ms$, the PLL unlocks on an average 10 times per second. For second-order PLLs, T_{av} has been found experimentally as a function of $(SNR)_L$.[1] The resulting curve is plotted in Fig. 2.46. To illustrate the theory, let us calculate a numerical example.

Numerical Example A second-order PLL is assumed to have the following specifications:

$$f_n = 10\,kHz$$
$$\omega_n = 62.8 \cdot 10^3\,s^{-1}$$
$$(SNR)_L = 1.5$$

From Fig. 2.46 we read $\omega_n T_{av} = 200$. Consequently $T_{av} = 3\,ms$. This means that the PLL unlocks about 300 times per second, which looks quite bad. However, the lock-in time T_L is found from Eq. (2.82) to be $T_L \approx 2\pi/\omega_n = 100\,\mu s$; that is, the PLL nevertheless is locked for 97 percent of the total time.

Summary of noise theory. Although the noise theory of the PLL is difficult, for practical design purposes it suffices to remember some rules of thumb:

1. Stable operation of the PLL is possible if $(SNR)_L$ is approximately 4.
2. $(SNR)_L$ is calculated from

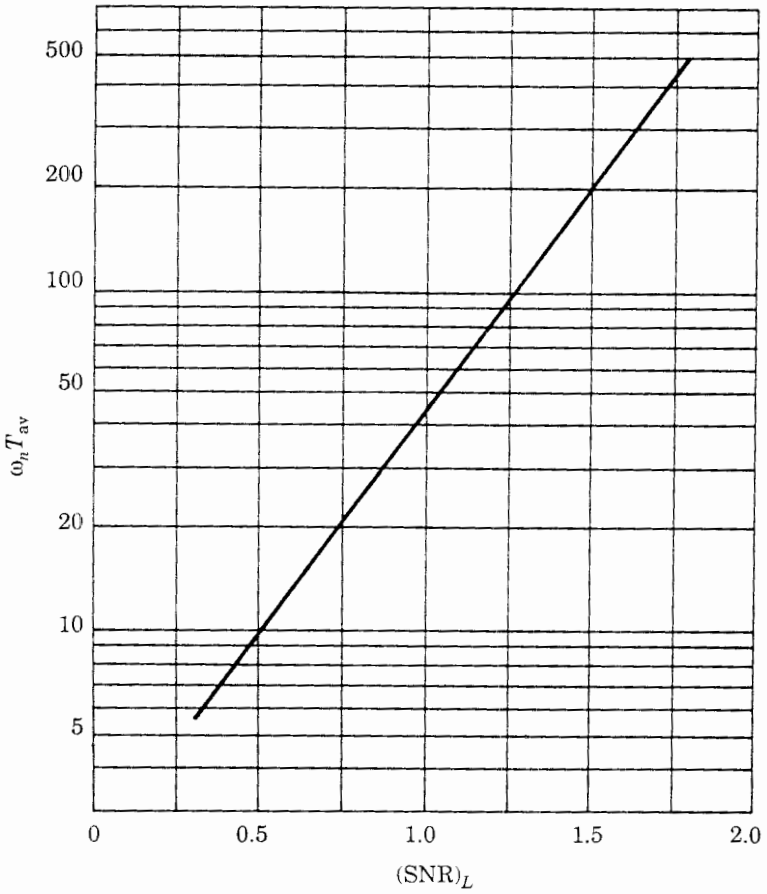


Figure 2.46 Average time T_{av} between two succeeding unlocking events plotted as a function of the $(SNR)_L$ at the output of the PLL. T_{av} is normalized to the natural frequency ω_n of the PLL. Note that this result is valid only for linear second-order PLLs. (*Adapted from Gardner*[1] *with permission.*)

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

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 (e.g., antenna, repeater)
B_L = noise bandwidth of the PLL

3. The noise bandwidth B_L is a function of ω_n and ζ. For $\zeta = 0.7$, $B_L = 0.53\ \omega_n$.
4. 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 that the statistical parameters of signal and noise do not change with time, i.e., that the power spectrum of noise remains always the same. In many applications of communication this condition is not met: the reference signal can almost disappear in some time intervals because of 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 has also some, though minor, influence.

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 (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 identical to either the down-scaled VCO output signal u_2' or 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 τ_1 for the active lead-lag filter. Hence the frequency created by the VCO will relatively slowly drift away toward the center frequency ω_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 were used, then the capacitor C (Fig. 2.16*c*) 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 remains nearly unchanged; thus the instantaneous output frequency of the VCO is held nearly constant. There is a good chance that the loop is still in lock when the reference signal reappears.

With the 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 type 1 or 2 is the better choice. Moreover, the active PI loop filter (type 3) offers the best performance relating to runaway of the VCO output frequency.

2.8.4 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 that 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, $(\text{SNR})_L$, 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_{sweep} (Fig. 2.47) over the frequency range of interest. Of course, the sweep rate must be held within the limits specified by Eq. (2.65); otherwise the PLL could not become locked. (In Sec. 2.4.3 we considered the case where the center frequency ω_0 was constant and the reference frequency ω_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 $\Delta\omega = \omega_1 - \omega_0'$ is the only parameter of importance.)

As shown in the block diagram of Fig. 2.47, 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. 2.47 to the HOLD position. To control the analog switch, we need a signal that tells us whether the PLL is in the locked state.

Such a control signal is generated by the in-lock detector shown in Fig. 2.47. The in-lock detector is a cascade connection of a 90° 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° between input signal u_1 and VCO output signal u_2. (*Note*: It is 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 $u_M = u_1'u_2$ 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 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 has to be added; this is not shown, however, in Fig. 2.47.

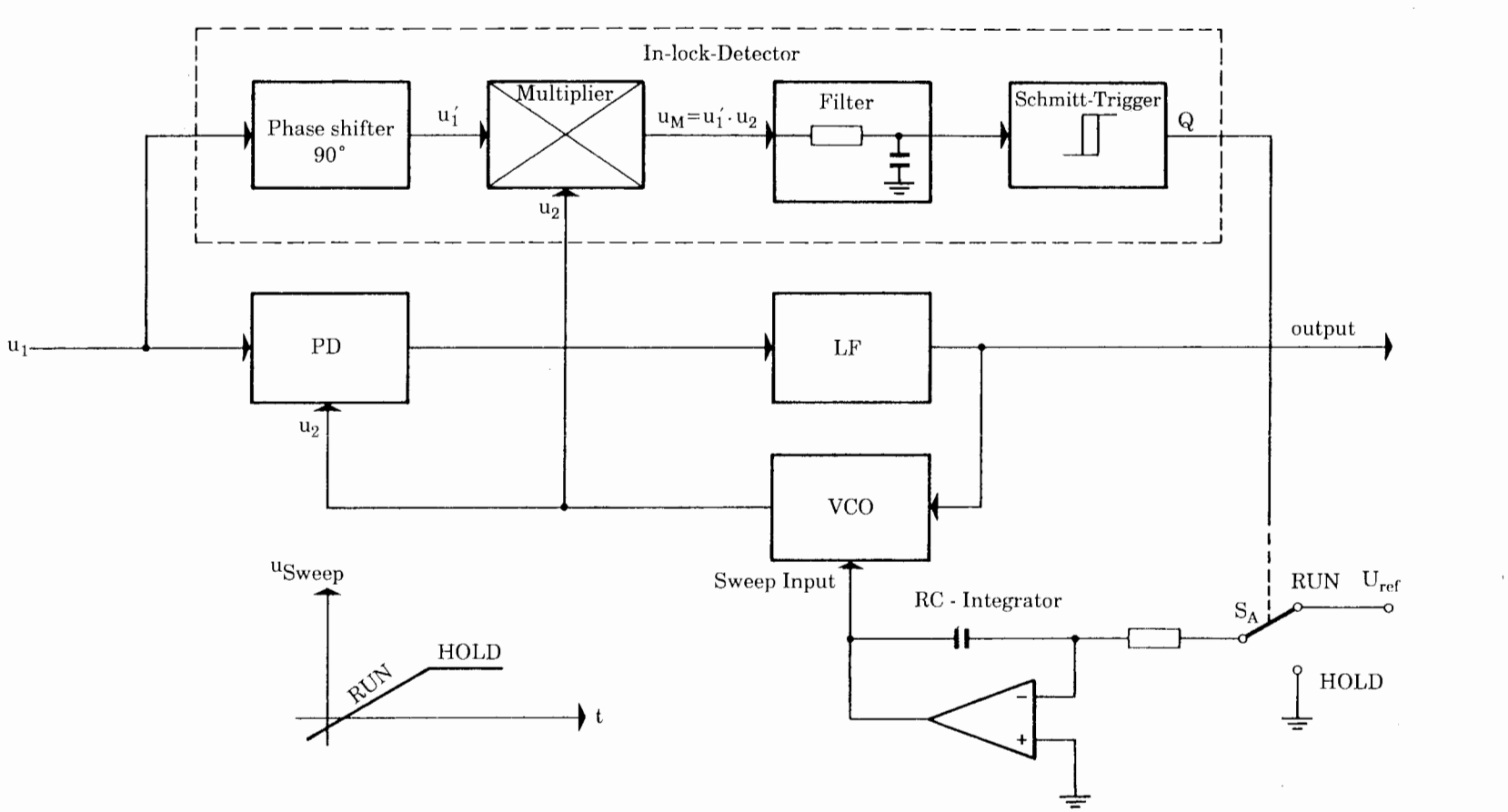


Figure 2.47 Simplified block diagram of a PLL using the sweep technique for the acquisition of signals buried in noise. (S_A = analog switch.)

The switched-filter technique. This locking method is depicted in Fig. 2.48. 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 that 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.

2.9 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.

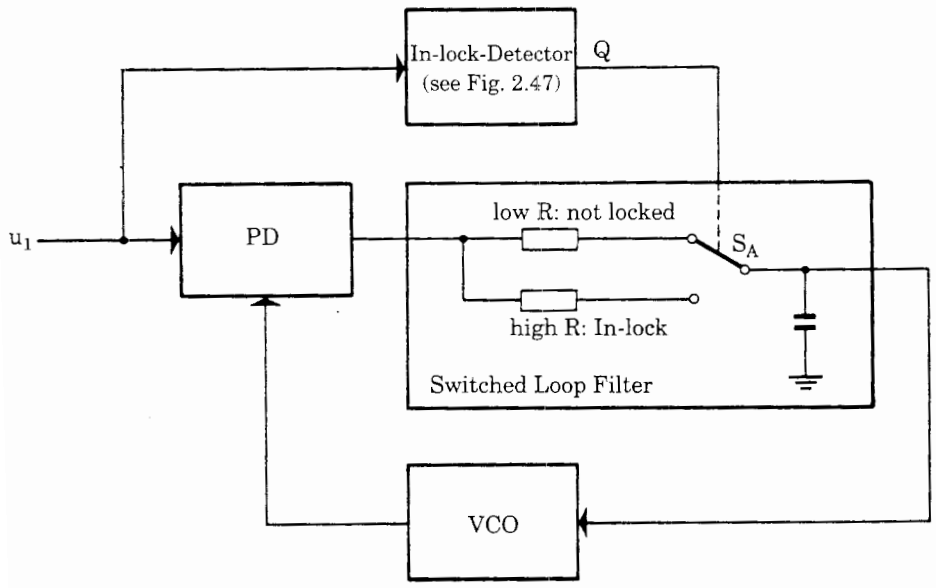


Figure 2.48 Simplified block diagram of a PLL using a switched loop filter for the acquisition of noisy signals. (S_A = analog switch.)

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 prcedure that is based on a flowchart, Fig. 2.49. To help in computing the relevant entities in the PLL design, the often used formulas for the PLL key parameters are listed in four tables, Tables 2.1 to 2.4. Table 2.1 shows the equations for hold range, lock range, etc., for a PLL using a type 1 phase detector. Table 2.2 presents the same information, but for an EXOR phase detector, etc.

The step-by-step design procedure of Fig. 2.49 should not be considered as 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 (e.g., 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 are based on the old industry standard CD4046 IC (from the 4000 CMOS series), which was originally introduced by RCA. (For details of PLL ICs refer to Table 10.1.) The individual steps are described in the following. As mentioned, most of the formulas used to design the PLL are listed in Tables 2.1–2.4.

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 (e.g., 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 scaler ratio must be determined. There are PLL applications where the output frequency f_2 always equals the reference frequency f_1. Here no down-scaler is needed; i.e., $N = 1$. There are cases 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 ($N_{\min} \le N \le N_{\max}$). When N is variable, natural frequency ω_n and damping factor ζ will vary with N, as seen from the corresponding equations in Tables 2.1 to 2.4. Both of these parameters will vary approximately with $1/\sqrt{N}$. Consequently ω_n will vary in the range $\omega_{n\,\min} < \omega_n < \omega_{n\,\max}$, and ζ will vary in the range $\zeta_{\min} < \zeta < \zeta_{\max}$. For these ranges we get approximately

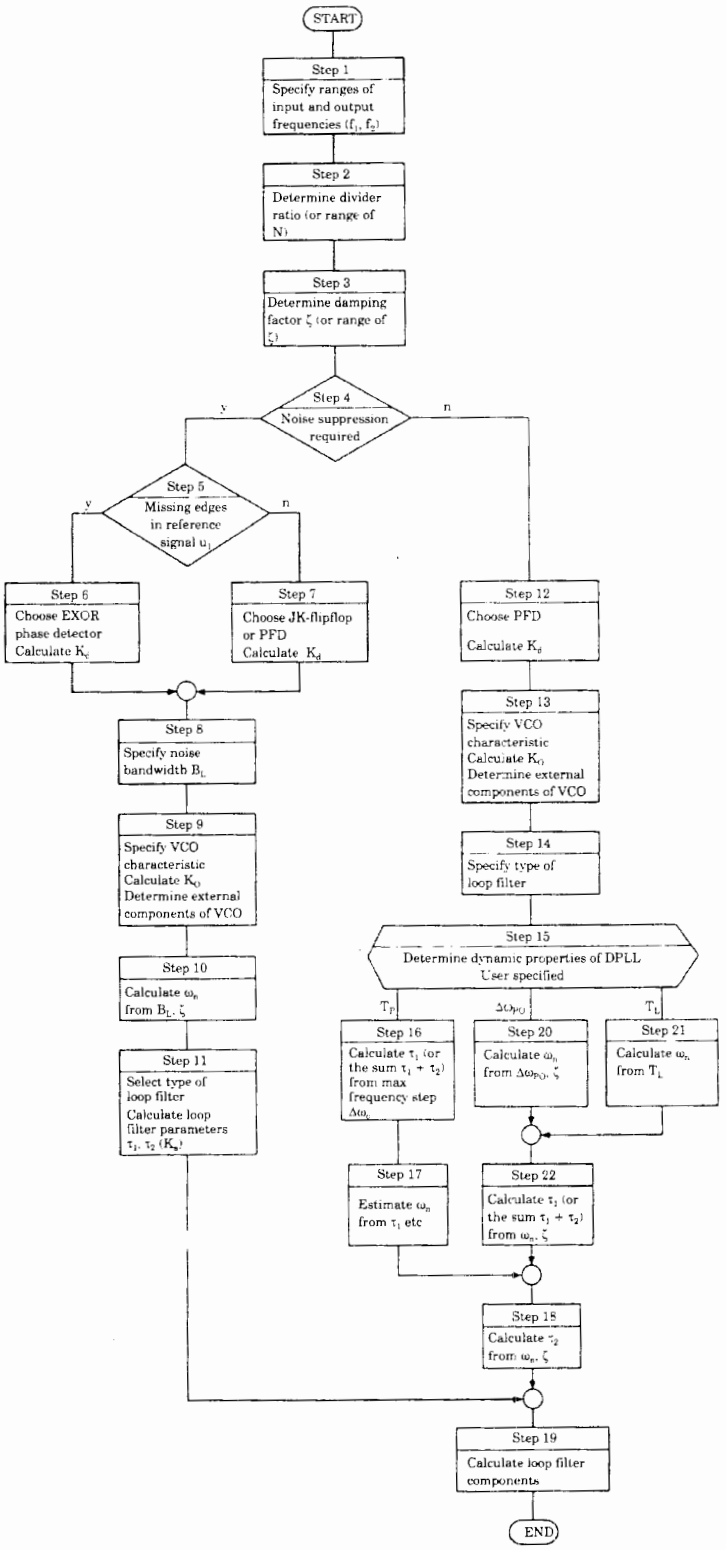


Figure 2.49 Flowchart of the mixed-signal PLL design procedure.

$$\frac{\omega_{n\max}}{\omega_{n\min}} = \sqrt{\frac{N_{\max}}{N_{\min}}} \tag{2.127}$$

and

$$\frac{\zeta_{\max}}{\zeta_{\min}} = \sqrt{\frac{N_{\max}}{N_{\min}}} \tag{2.128}$$

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, ζ varies by about a factor of 3, which can be tolerated.

Much larger variations of ζ, however, have to be avoided, because the loop then would get oscillatory for the smallest and sluggish for the largest damping factor. When N varies over a large range (e.g., 1 to 100), it is often 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 ζ becomes optimum for a divider ratio N_{mean}, which is given by the geometric mean of $N_{\max}$ and $N_{\min}$,

$$N_{\text{mean}} = \sqrt{N_{\min} N_{\max}} \tag{2.129}$$

(For constant N, $N_{\text{mean}} = N$, of course.) If $N_{\min} = 10$ and $N_{\max} = 100$, for example, N_{mean} would be $31.6 \rightarrow 32$. Choosing $\zeta = 0.7$ for $N = 32$ would yield a minimum damping factor of $\zeta_{\min} = 0.4$ and a maximum of $\zeta_{\max} = 1.2$, which is a fair compromise.

Step 3. Determine the damping factor ζ. When N is constant, ζ remains constant too and can be chosen arbitrarily. For constant N, it is optimum to select $\zeta = 0.7$; the PLL then has the Butterworth response, as explained in Sec. 2.4.2. 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 whether the PLL should offer noise suppression or not. If a digital frequency synthesizer has to 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 it continues at step 12.

Step 5. Noise must be suppressed by this PLL. As discussed in Sec. 2.8.4, 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 it continues 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 which 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. 2.3.1, the PFD offers superior performance, e.g., infinite pull-in range, so this type of phase detector is normally preferred. 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. 2.3.1 should be used. The procedure continues at step 8.

Step 8. The noise bandwidth B_L must now be specified. As shown in Sec. 2.8.4, B_L is related to the signal-to-noise ratio of the loop $(\mathrm{SNR})_L$ by

$$(\mathrm{SNR})_L = (\mathrm{SNR})_i \frac{B_i}{2B_L} \tag{2.130}$$

B_L should be chosen such that $(\mathrm{SNR})_L$ becomes larger than some minimum value, typically larger than 4 (which corresponds to 6 dB). If $(\mathrm{SNR})_i$ is not known, it must be estimated or eventually measured.[17] Finally, the noise bandwidth B_i at the input of the PLL must also be known. B_i is nothing other than the bandwidth of the signal source or the bandwidth of an optional prefilter. After $(\mathrm{SNR})_i$ and B_i are determined, B_L can be calculated from Eq. (2.130).

An additional problem arises when the scaler ratio N must be variable. We know from noise theory that B_L is also related to ω_n and ζ by Eq. (2.120), which reads

$$B_L = \frac{\omega_n}{2}\left(\zeta + \frac{1}{4\zeta}\right) \tag{2.131}$$

When the divider ratio N is variable, both ω_n and ζ 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 that B_L does not fall below the minimum acceptable value, we should check its values at the extremes of the scaler ratio N, using Eq. (2.131). 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. Because the center frequency ω_0 (or the range of ω_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 $U_B = 5\text{V}$ 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. 2.50. Now the VCO gain K_0 is calculated from the slope of this curve, i.e.,

$$K_0 = \frac{\omega_{2\max} - \omega_{2\min}}{u_{f\max} - u_{f\min}}$$

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.

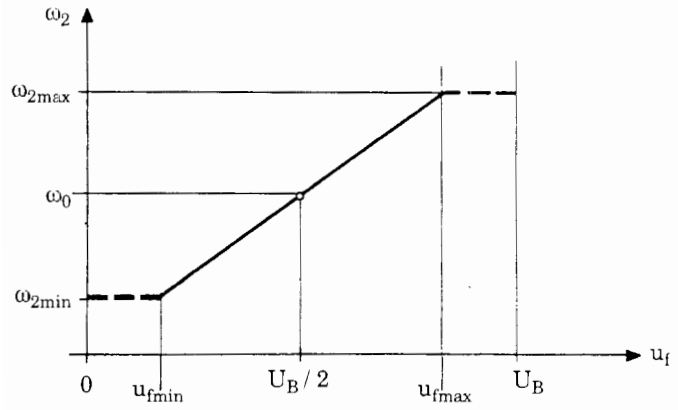


Figure 2.50 Characteristic of the VCO. The curve shows angular frequency ω_2 versus control voltage u_f. Note that the useful voltage range is less than the supply voltage.

Step 10. Calculate the natural frequency ω_n. Given noise bandwidth B_L and damping factor ζ, ω_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 ζ have been determined for $N = N_{\text{mean}}$; thus the value obtained for ω_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. Given ω_n, ζ, K_0, K_d, and N, the equations for ω_n and ζ in Tables 2.1 to 2.4 are used to calculate the two time constants τ_1 and τ_2, respectively. 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 from 2 to 10. 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. When the PFD 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. 2.3.1.

Step 13. In this step the characteristic of the VCO will be determined (refer also to Fig. 2.50). Because the center frequency ω_0 (or the range of ω_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 $U_B = 5\,\text{V}$ 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. 2.50. Now the VCO gain K_0 is calculated from the slope of this curve, i.e.,

$$K_0 = \frac{\omega_{2\,\max} - \omega_{2\,\min}}{u_{f\,\max} - u_{f\,\min}}$$

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 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 "Pull-in Range $\Delta\omega_P$ and Pull-in Time T_P" in Sec. 2.6.2. Other types of loop filters do not bring significant benefits.

Step 15. Determine the dynamic properties of the PLL. To select an appropriate value for the natural frequency ω_n, we must have an idea of how the PLL should react on dynamic events, e.g., 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; i.e., there are (at least) three different ways of specifying dynamic performance of the PLL.

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_{ref}; i.e., the frequency at the output of the VCO is given by $f_2 = N f_{\text{ref}}$, 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," i.e., if f_2 changes from $N_0 f_{\text{ref}}$ to $(N_0 + 1) f_{\text{ref}}$. In this case, the pull-out range $\Delta\omega_{PO}$ is required to be less than f_{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 to develop such a specification 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. Given the maximum pull-in time T_P allowed for the greatest frequency step at the output of the VCO, the time constant τ_1 (or the sum of both time constants $\tau_1 + \tau_2$) of the loop filter is calculated using the formula for T_P given in Tables 2.1 to 2.4. When the loop filter is a passive lead-lag, we can determine

only the sum $\tau_1 + \tau_2$. For the other types of loop filters, τ_1 can be computed directly. For $\Delta\omega_0$ the maximum (angular) frequency step at the VCO output must be entered. The design proceeds with step 17.

Step 17. When the loop filter is a passive lead-lag, we computed the sum $\tau_1 + \tau_2$ in step 16. Hence we can calculate ω_n immediately from the corresponding equation in Tables 2.1 to 2.4. For the other loop filter types, τ_1 was computed in step 16. ω_n can be computed now by using the appropriate equation in Tables 2.1 to 2.4. The design proceeds with step 18.

Step 18. Given ω_n and ζ, now the time constant τ_2 has to be calculated by using the formula for ζ given in Tables 2.1 to 2.4. When the sum $\tau_1 + \tau_2$ has been computed in step 16, τ_1 can now be determined. When a passive lead-lag filter is used, strange things may happen at this point: we perhaps obtained $\tau_1 + \tau_2 = 300\,\mu s$ in a previous step and computed $\tau_2 = 400\,\mu s$ right now! To realize the intended system, we would have to choose $\tau_1 = -100\,\mu s$, which is impossible. Whenever we meet that situation, the system cannot be realized with the desired goals (i.e., with the desired values for ω_n and/or ζ). It only becomes realizable if we choose a lower ω_n, for example, which increases $\tau_1 + \tau_2$, or if we specify a lower ζ, which decreases τ_2.

Step 19. Given τ_1 and τ_2 (plus eventually K_a), the components of the loop filter can be determined. If the passive lead-lag filter is chosen, we have $R_1C = \tau_1$ and $R_2C = \tau_2$. C can be chosen arbitrarily; it should be specified such that we get "reasonable" values for R_1 and R_2, i.e., in the range of kilohms to megohms. For the active lead-lag filter, we have $K_a = C_1/C_2$, $\tau_1 = R_1C_1$, and $\tau_2 = R_2C_2$. Here we should first specify C_1 so as 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 $\tau_2 = R_2C_2$. For the active PI filter, we have $R_1C = \tau_1$ and $R_2C = \tau_2$. As with the passive lead-lag filter, we would first select C to get reasonable values for R_1 and R_2.

Step 20. Given pull-out range $\Delta\omega_{PO}$ and damping factor ζ, the formula for $\Delta\omega_{PO}$ given in Tables 2.1 to 2.4 is used to calculate the natural frequency ω_n. The design proceeds with step 22.

Step 21. Given the lock-in time T_L, the natural frequency ω_n is calculated from the formula for T_L in Tables 2.1 to 2.4. The procedure continues with step 22.

Step 22. Given ω_n, the time constant τ_1 can be calculated by using the formula for ω_n in Tables 2.1 to 2.4, when the loop filter is either an active lead-lag or a PI filter. When the passive lead-lag filter is used, however, only the sum $\tau_1 + \tau_2$ can be computed from the formula for ω_n. In that case, the final value for τ_1 will be known after τ_2 has been calculated in step 18. The design proceeds with step 18.

Though the design procedure seems quite complex, it would be premature to consider it universal. Some more special parameters are not at all taken into account, such as spectral purity of the output signal of frequency synthesizers and related quantities. We have to consider such effects when discussing the most important PLL applications.

A practical design example that makes use of the design procedure will be presented in Sec. 3.4.

2.10 Mixed-Signal PLL Applications

One of the most important applications of the mixed-signal PLL is frequency synthesis. In the design of PLL frequency synthesizers we are confronted with a number of problems that are specific for that topic. For this reason PLL frequency synthesizers will be discussed in a separate chapter (Chap. 3). In this section we are going to consider some other typical PLL applications.

2.10.1 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; i.e., 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.

There are basically two different principles of transmitting digital data, namely, *baseband* 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.[20] 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. 9.

In this section we deal only with baseband systems. Let us assume that a digital signal, i.e., 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 RS-485 standard.[28] 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, e.g., 5 V and 0 V, respectively. This simplest data format is sketched in the first row of Fig. 2.51 and is referred to as *NRZ code* (NRZ = *n*on *r*eturn to *z*ero). 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. Such a character mostly consists of a series of 8 data bits. The data bits are headed by a start

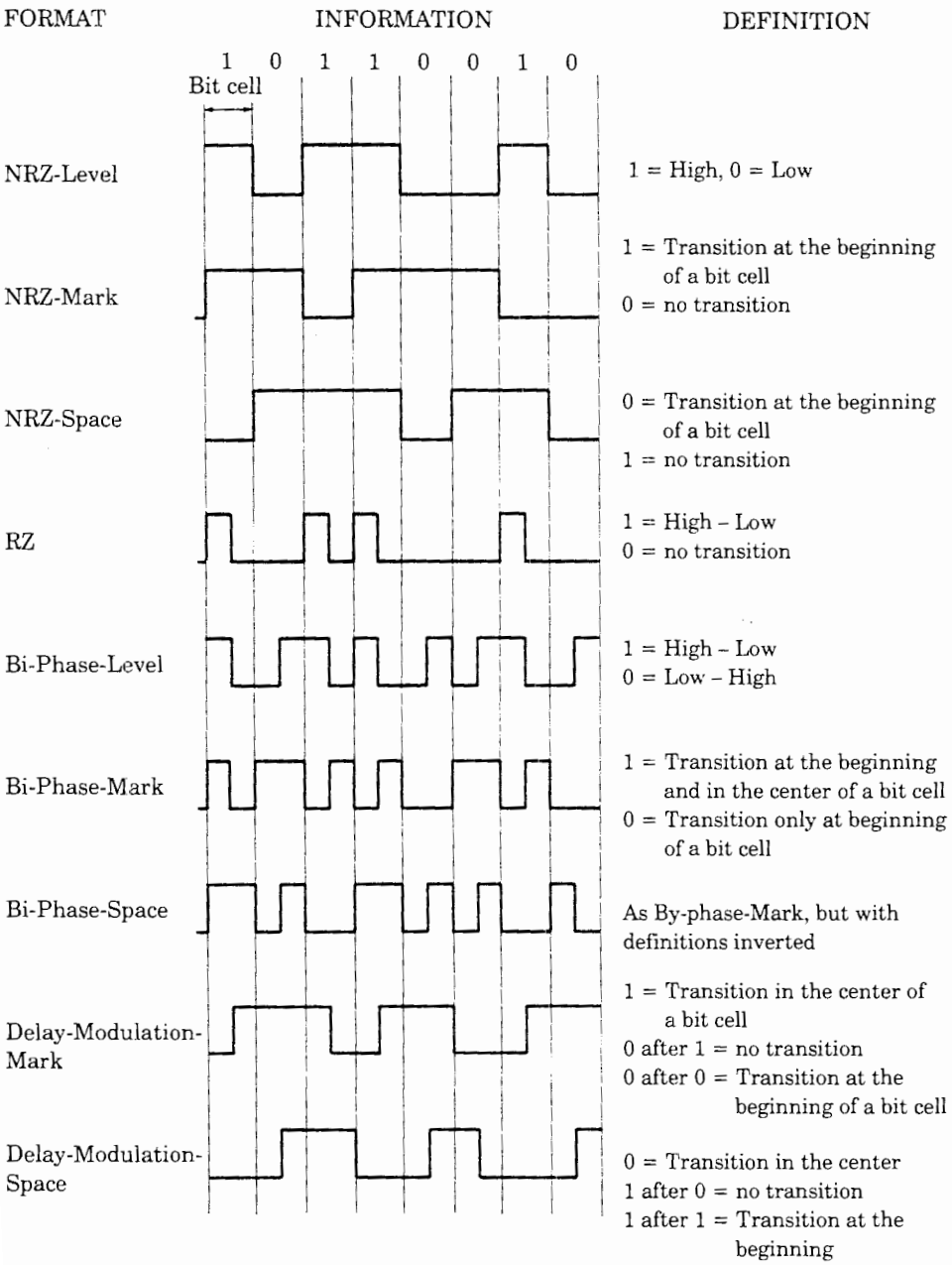


Figure 2.51 Review of the most commonly used binary and pseudo-ternary formats.

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 were used and say 1000 ones 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 ones have been sent.

To provide more clocking information, a number of other codes have been devised, as shown in Fig. 2.51. 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 readily 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 shows 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 in the fact that the required bandwidth to transmit a given number of bits per second is twice the bandwidth of the NRZ code.[20]

There is another code, called the *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 becomes evident that the methods of extracting clock information must depend on the particular code used. Let us have a look at the most important recovery techniques.

Let us start with recovering the clock signal for the RZ format (Fig. 2.51). The corresponding block diagram is shown in Fig. 2.52. 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. 2.52. If an EXOR PD is used, the VCO generates a square wave signal u_z^*, which is in quadrature to the demodulated data signal, as seen from the waveforms in Fig. 2.52. 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 have to be

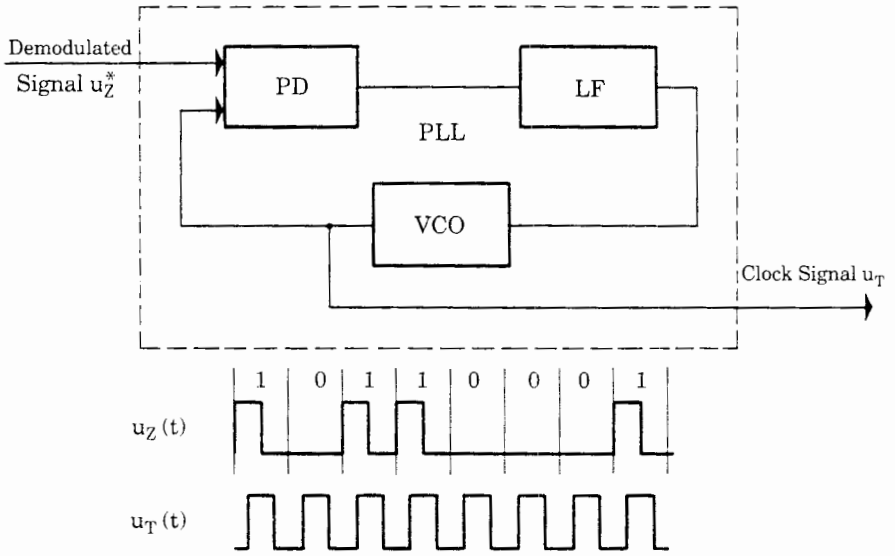


Figure 2.52 Operating principle for clock recovery in the RZ format.

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 9 bits there is *at least 1 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 that 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 must be reestablished; this is done by 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.[1,20]

One method of extracting the clock frequency from the NRZ signal is shown in Fig. 2.53. The demodulated NRZ signal u_z^* is first differentiated. Then, the differentiated signal u_{diff} is rectified by an absolute-value circuit.[20,34] As seen in the waveforms in Fig. 2.53, the rectified signal contains a frequency component synchronous with the clock. Hence it can be used to synchronize a PLL; the 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.

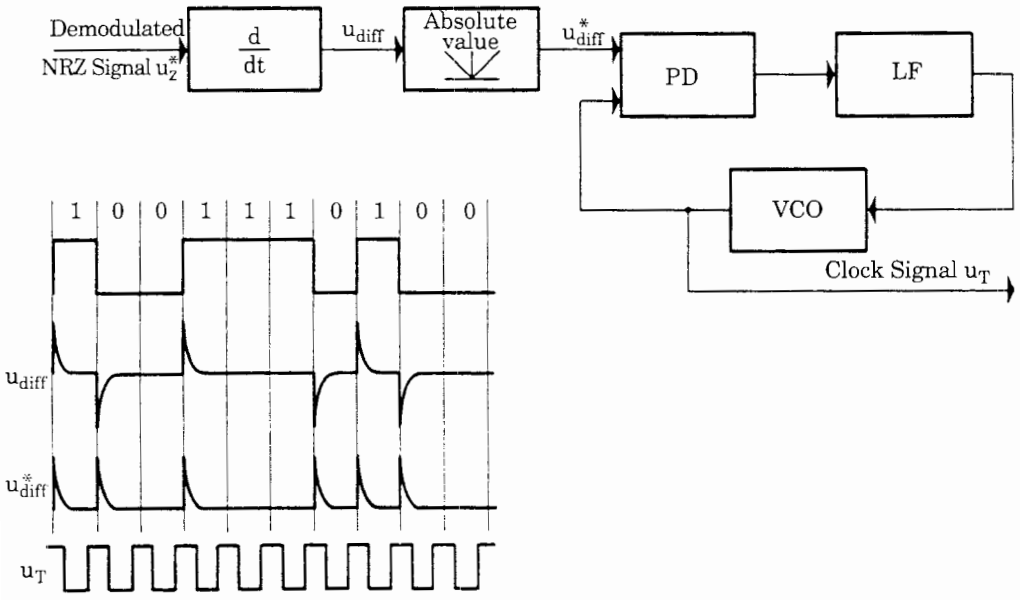


Figure 2.53 Operating principle for clock recovery in the NRZ format.

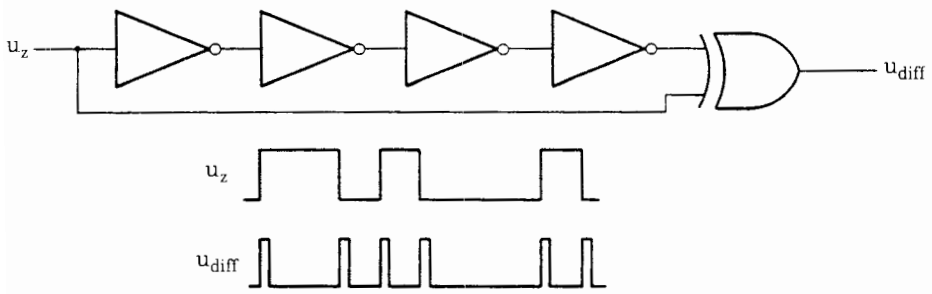


Figure 2.54 The function of an analog differentiator followed by an absolute-value circuit can be alternatively performed by a digital edge detector.

Figure 2.54 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 the beginning and in the center of a bit cell, as pointed out in Fig. 2.51. As in the circuits in Figs. 2.52 and 2.53, 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. 2.55.[27] The waveforms produced by this device are depicted in Fig. 2.56.

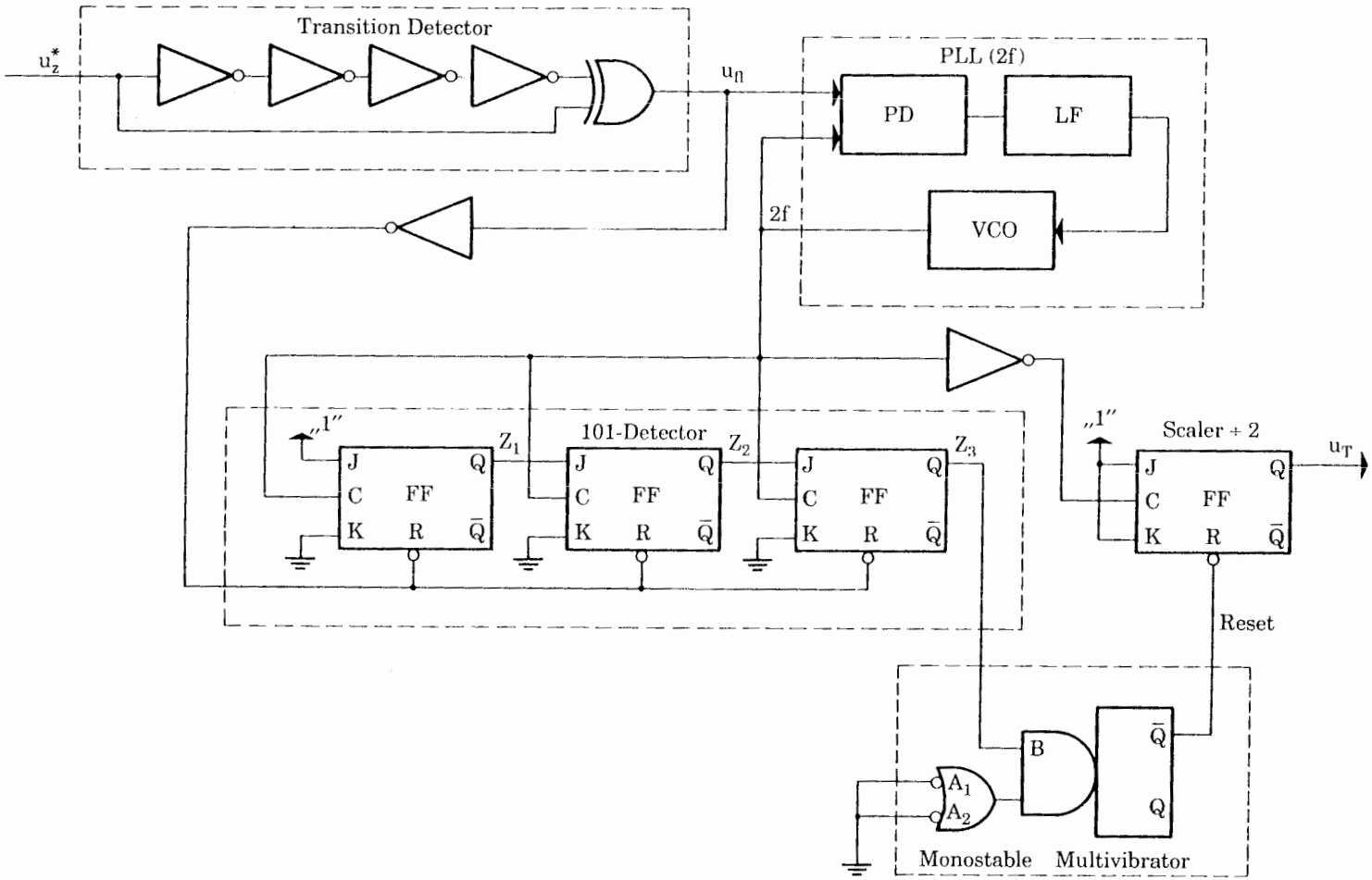


Figure 2.55 Circuit for clock signal recovery with the DM (delay modulation) format. All flipflops are triggered onto the negative-going edge of the clock signal C. Input B of the monostable multivibrator triggers on the positive-going edge; inputs A_1 and A_2 trigger on the negative-going edge. Inputs A_1 and A_2 are unused here.

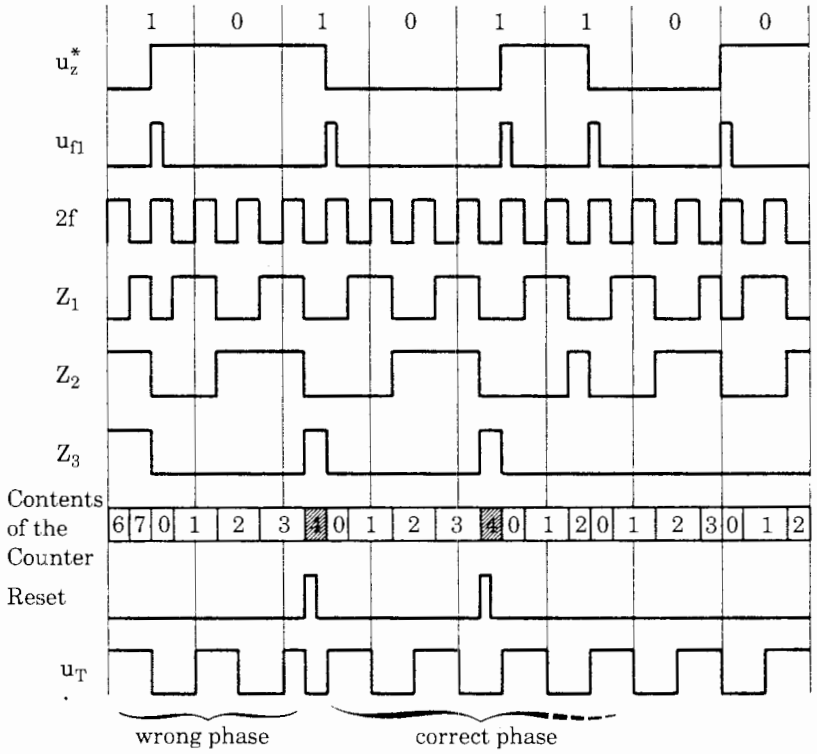


Figure 2.56 Waveforms of the circuit of Fig. 2.55.

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. 2.56). The output signal of the VCO ($2f$) is scaled down in frequency by a factor of 2; the right-most JK-flipflop of Fig. 2.55 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. 2.55 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. 2.56. An additional circuit is needed, which takes corrective action. Consider again the signal u_{f1} in Fig. 2.56. 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 2 periods is impossible. If a three-stage binary UP counter (labeled as the 101 detector in Fig. 2.55) is used to count the (negative) transitions of the signal $2f$, and if this counter is reset by every u_{f1} pulse, its content never exceeds 4. Moreover, as we can clearly see in

Fig. 2.56, the content of 4 can be obtained only in the first half of a bit cell, never in the second half.

This fact is used to properly reset the divide-by-2 flipflop in Fig. 2.55. Whenever the 3-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 seen from the three examples discussed, the PLL is the key element in each of the applications.

2.10.2 Motor-speed control

Very precise motor-speed controls at low cost are possible with 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. 2.57. 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 2.58 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. 2.58.

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. 2.59. 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

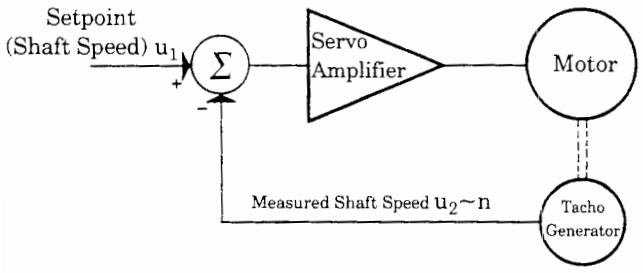


Figure 2.57 Block diagram of a conventional motor-speed control system.

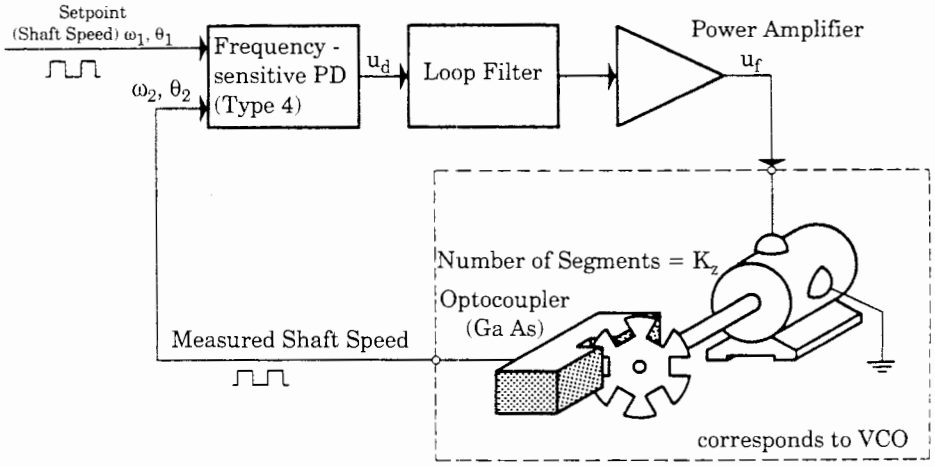


Figure 2.58 Block diagram of a motor-speed control system based on PLL techniques.

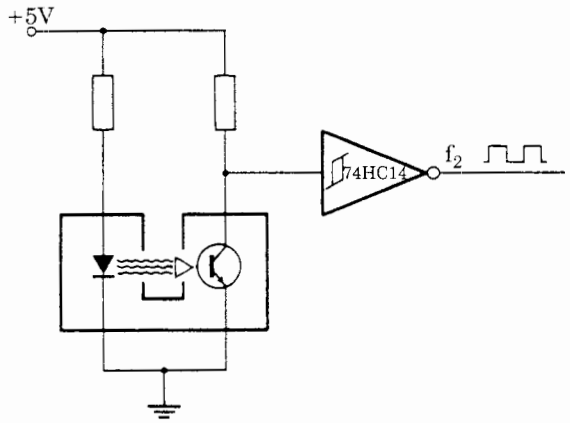


Figure 2.59 Circuit of the optical tachometer generator shown in Fig. 2.58.

phototransistor stage is normally followed by a Schmitt trigger, such as a 74HC/HCT14-type CMOS device.

The signal generated by the optocoupler has a frequency proportional to the speed of the motor. Because the phase detector compares not only the frequencies ω_1 and ω_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 phase detector.

To analyze the stability of the system, the transfer functions of all blocks in Fig. 2.58 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

$$\omega(t) = K_m u_f \left[1 - \exp\left(-\frac{t}{T_m}\right)\right] \tag{2.132}$$

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. 2.60. Equation (2.132) indicates that ω will settle at a value proportional to u_f after some time. Applying the Laplace transform to Eq. (2.132) yields

$$\Omega(s) = U_f(s)\frac{K_m}{1+sT_m} \tag{2.133}$$

The phase angle ω of the motor is the time integral of the angular speed ω. Therefore, we get for its Laplace transform $\Phi(s)$ the expression

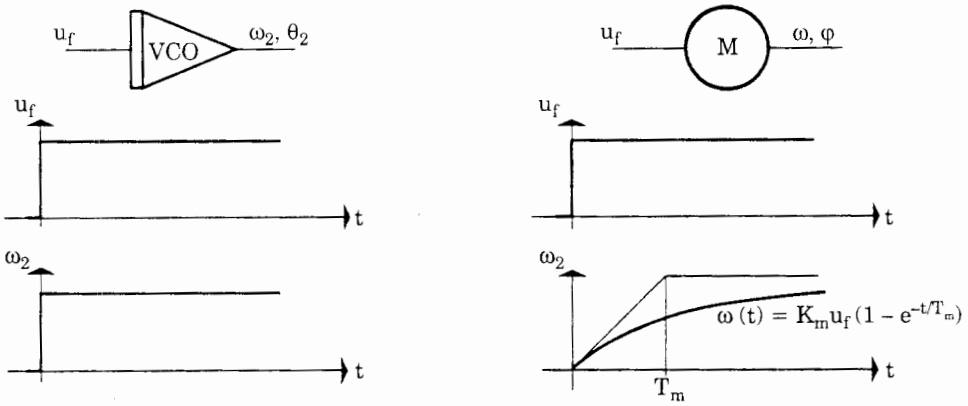
$$\Phi(s) = U_f(s)\frac{K_m}{s(1+sT_m)} \tag{2.134}$$

The sector disk shown in Fig. 2.58 has K_Z teeth. This implies that the phase of the tachometer signal is equal to phase ϕ multiplied by K_Z. Consequently we obtain for $\Theta_2(s)$

$$\Theta_2(s) = \frac{K_m K_Z}{s(1+sT_m)}U_f(s) \tag{2.135}$$

The transfer function $H_m(s)$ of the motor is therefore given by

$$H_m(s) = \frac{K_m K_Z}{s(1+sT_m)} \tag{2.136}$$



$$\Omega_2(s) = K_o U_f(s) \qquad \Omega(s) = \frac{K_m U_f(s)}{1+sT_m}$$

$$\Theta_2(s) = \frac{K_o U_f(s)}{s} \qquad \Phi(s) = \frac{K_m U_f(s)}{s(1+sT_m)}$$

Figure 2.60 The step response of an ordinary VCO compared with that of a motor. Note that the VCO is a first- and the motor a second-order system.

The motor is evidently a second-order system, whereas the VCO [according to Eq. (2.34)] was a first-order system only. In Fig. 2.60 the transient response of the motor is compared with that of an ordinary VCO. The motor-speed control system of Fig. 2.58 is therefore a third-order system. The mathematical model of the control system can now be plotted (Fig. 2.61). The servo amplifier is supposed to be a zero-order gain block with proportional gain K_a. The poles of this amplifier normally can 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° 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. 2.61 can be combined into fewer blocks, a result which yields the simpler block diagram of Fig. 2.62. 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 τ_2) then have to be chosen for best dynamic performance and maximum stability of the system. There are many ways to solve this problem. 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 by the Bode diagram.[3] In the Bode diagram, amplitude and phase response of the open-loop gain of the system are plotted. From Fig. 2.63, the open-loop gain $G_0(s)$ is given by

$$G_0(s) = G(s)H(s) = K\frac{1+s\tau_2}{s^2(1+sT_m)} \tag{2.137}$$

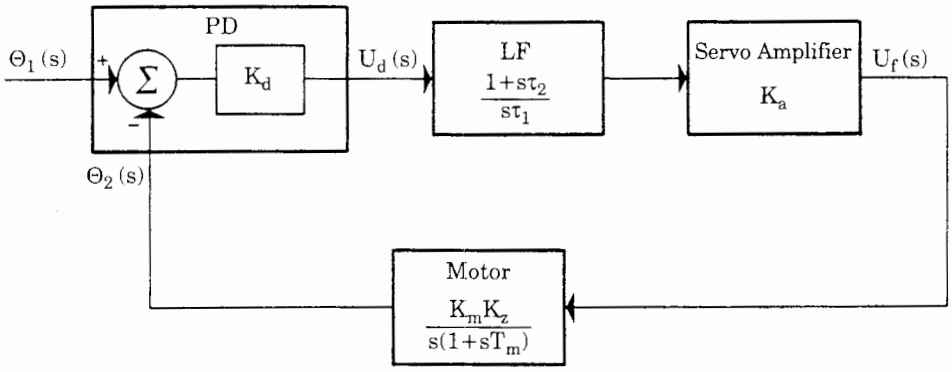


Figure 2.61 Mathematical model of the motor-speed control system of Fig. 2.58.

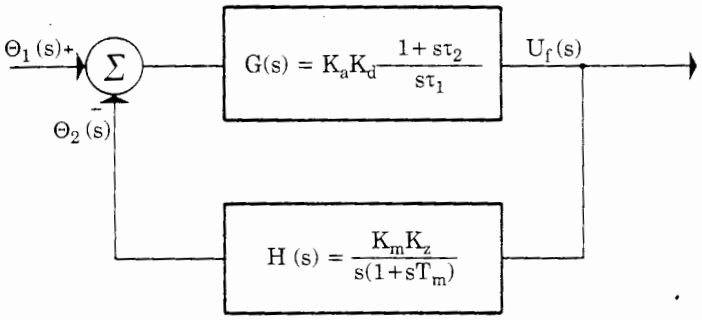


Figure 2.62 Condensed mathematical model of the motor-speed control system.

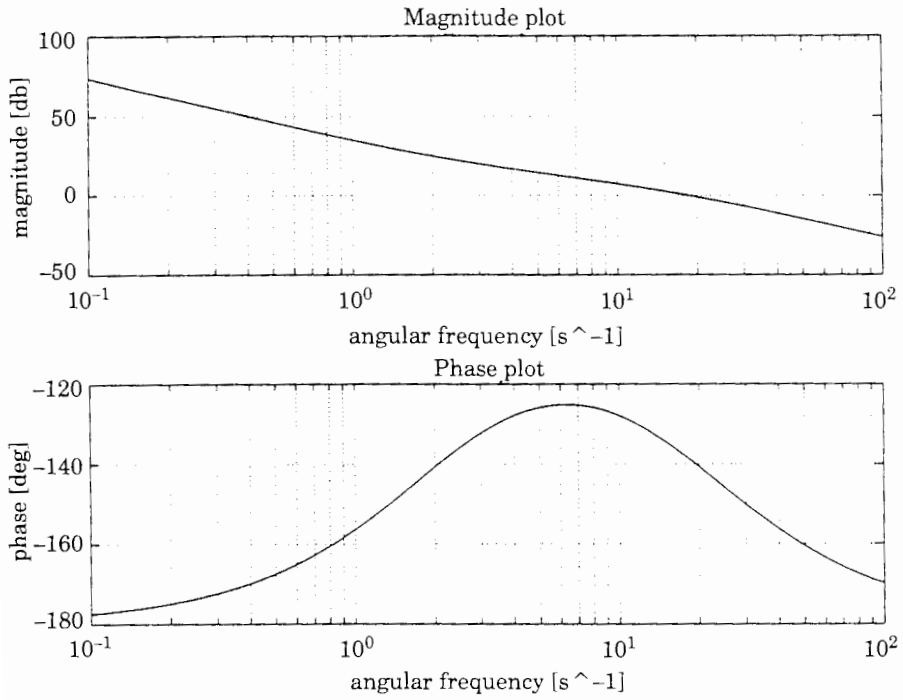


Figure 2.63 Bode diagram for the open-loop gain of the motor-speed control system according to Fig. 2.58.

where K contains the gain factors of the individual blocks,

$$K = \frac{K_d K_a K_m K_Z}{\tau_1} \tag{2.138}$$

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 τ_1 and τ_2 can be chosen freely. We assume here that

$$T_m = 0.05\,\text{s}$$
$$K_m = 1$$

We use MathWorks Control System Toolbox[26] to plot and optimize the Bode diagram of our control system.[25] Looking at Eq. (2.137), 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 angular frequency $\omega_m = 1/T_m$, the slope will become –60 dB/decade, so at higher frequencies the phase would approach 270°, 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° 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*[3] of at least 30°. 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, τ_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. 2.63. The magnitude curve starts with –40 dB/decade at low frequencies. The transfer function zero at $\omega = 1/\tau_2 = 2\,\text{s}^{-1}$ causes the slope to change to –20 dB/decade; i.e., the magnitude curve bends up. This causes the phase curve to increase from –180° toward –90°, as is seen from the phase plot. At the break frequency $\omega = 1/T_m = 20\,\text{s}^{-1}$, the slope of the magnitude curve becomes –40 dB/decade again. The magnitude crosses the 0-dB line at about $\omega = 20\,\text{s}^{-1}$, and the phase margin is about 40°, which is sufficient for good transient response. Knowing total gain K, the amplifier gain K_a and the remaining time constant τ_1 can be specified as soon as the number of teeth K_Z is given.

Chapter 3

3

PLL Frequency Synthesizers

3.1 Synthesizers in Wireless and RF Applications

PLL frequency synthesizers play an ever-increasing role in the field of communications. Originally, the frequency synthesizer was 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, 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 the fractional part of an arbitrary number. Whereas fractional-N synthesizers have been considered rather exotic in the past, they suddenly have gained increased interest, mainly in spread-spectrum applications.

Conventional communications used one single carrier whose frequency was fixed. Radio and TV transmitters are examples for this category. In military communications it turned out that such constant-carrier-frequency links could easily be corrupted by "jammers."[20] This led 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 that 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 vocabulatory. The duration of such a chip is usually in the order of several hundred microseconds. This implies that the receiver must be able to switch the carrier frequency very fast, i.e., within about 100 µs. It will be demonstrated in the next section that fractional-N synthesizers can switch their output frequency more rapidly than conventional integer-N synthesizers. Frequency hopping has gained increased importance in civil communications, since it is used in the newer generations of mobile phones.[49]

3.2 PLL Synthesizer Fundamentals

This section is subdivided into two subsections:

- Integer-N frequency synthesizers
- Fractional-N frequency synthesizers

Fractional-N synthesizers are based on the simpler integer-N synthesizers, but include a number of additional circuits. Hence we start the discussion with the integer-N systems.

3.2.1 Integer-*N* frequency synthesizers

A very simple frequency synthesizer has already been shown in Fig. 2.1. This circuit is redrawn in Fig. 3.1, where it is shown that the scaling factor N is usually set by an external digital signal. In Fig. 3.1 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 f_1. In this simple arrangement the VCO creates an output frequency f_2 which is simply Nf_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 on the chip, as shown in Fig. 3.2.

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

$$f_2 = \frac{N}{R} f_{osc} = Nf_1 \tag{3.1}$$

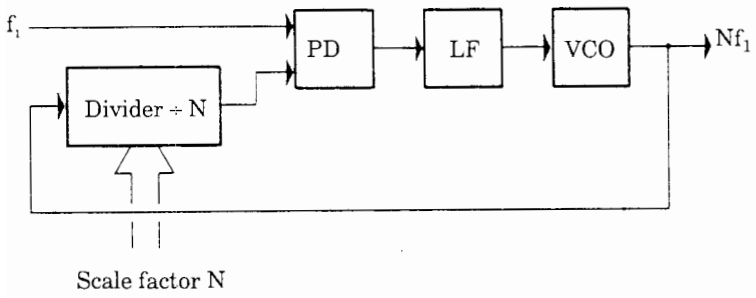


Figure 3.1 Basic frequency synthesizer system.

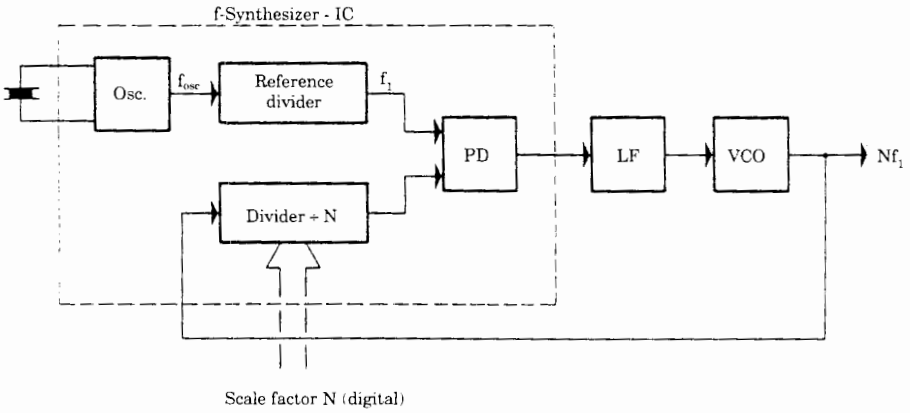


Figure 3.2 System equal in performance to Fig. 3.1 but with an additional reference divider that makes it possible to use a higher-frequency reference, normally a quartz oscillator.

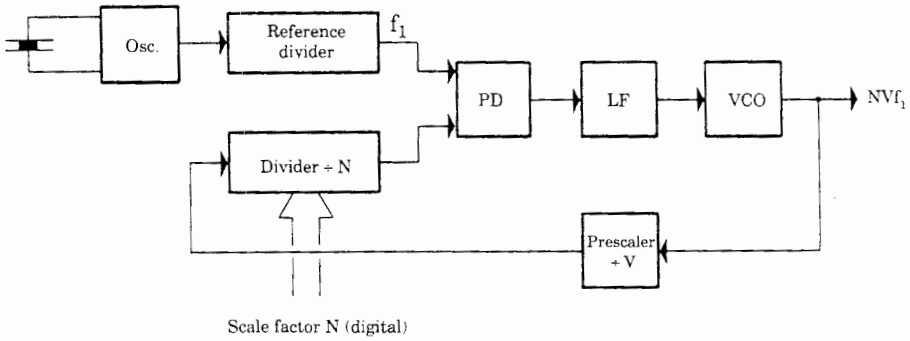


Figure 3.3 Frequency synthesizer extends the upper frequency range by using an additional high-speed prescaler. The channel spacing is increased to Vf_1, where f_{osc} is the frequency of the oscillator.

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, phase detectors, frequency dividers, and so on, as indicated by the dashed enclosure in Fig. 3.2. Because of 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). To generate higher frequencies, *prescalers* are used; these are built with other IC technologies such as ECL, Schottky TTL, GaAs (gallium arsenide), or SiGe (silicon-germanium compound) (Fig. 3.3). Such prescalers extend the range of frequencies into the microwave frequency bands.[7,37]

If the scaling factor of the prescaler is V (Fig. 3.3), the output frequency of the synthesizer becomes

$$f_{out} = NVf_1$$

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 $10f_1$, $20f_1$, $30f_1$, . . . can be generated. This disadvantage can be circumvented by using a so-called dual-modulus prescaler, as shown in Fig. 3.4.[11,38]

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. 3.4 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. 3.4:

1. Both programmable $\div N_1$ and $\div N_2$ counters are downcounters.
2. The output signal of both of these counters is high if the content of the corresponding counters has not yet reached the value 0.
3. 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.
4. N_1 is always greater than or equal to N_2.
5. As shown by the AND gate in Fig. 3.4, 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. 3.4 becomes clearer if we assume that the $\div N_1$ counter has just counted down to 0 and both counters have just been loaded with their preset values N_1 and N_2, respectively. We now have to

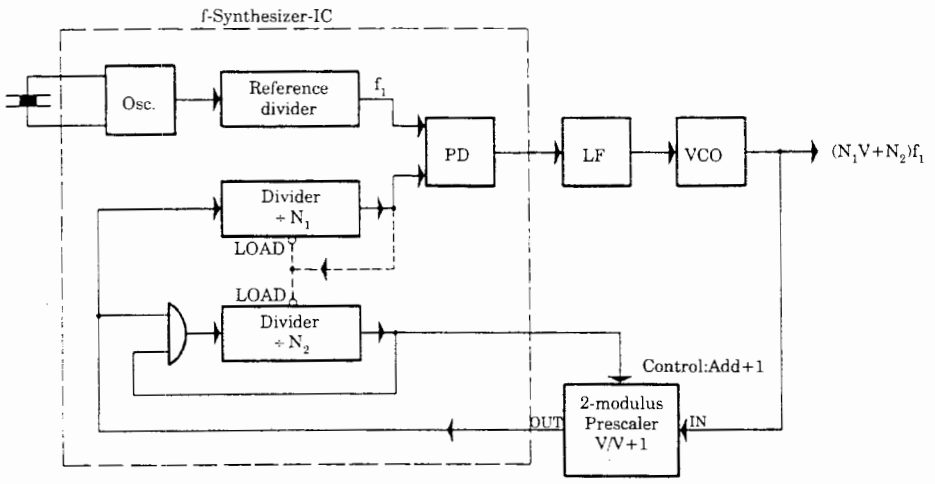


Figure 3.4 Frequency synthesizer using a dual-modulus prescaler. The channel spacing becomes f_1.

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_{tot} of the arrangement shown in Fig. 3.4. 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 $N_2(V + 1)$ 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 will step 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_{tot} did the VCO produce in order to run through one full cycle? N_{tot} is given by

$$N_{tut} = N_2(V+1) + (N_1 - N_2)V$$

Factoring out yields the simple expression

$$N_{tut} = N_1 V + N_2 \tag{3.2}$$

As stated above, 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. (3.2) becomes

$$N_{tot} = 10N_1 + N_2 \tag{3.3}$$

In this expression, N_2 represents the units and N_1 the tens of the overall division ratio N_{tot}. Then N_2 must be in the range of 0 to 9, and N_1 can assume any value greater than or equal to 9; that is, $N_{1min} = 9$. The smallest realizable division ratio is therefore

$$N_{tot,min} = N_{1\,min} V = 90$$

The synthesizer of Fig. 3.4 is thus able to generate all integer multiples of the reference frequency f_1 starting from $N_{tot} = 90$.

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

$$N_{tot} = 16N_1 + N_2$$

Now N_2 would be required to have a range of 0 to 15, and the minimum value of N_1 would be $N_{1min} = 15$. In this case, the smallest realizable division ratio $N_{tot,min}$ 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. 3.4 is 10 kHz. The smallest output frequency would then be $90f_1 = 900$ 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 with 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. For $V = 100$, ratio N_{tot} would be

$$N_{tot} = 100N_1 + N_2$$

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 $N_{tot,min}$ is no longer 90 but has been increased to

$$N_{tot} = 100N_{1\,min} = 100 \cdot 99 = 9900$$

If the reference frequency f_1 is still 10 kHz, the lowest frequency to be synthesized is now 99 MHz.

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. 3.5). The four-modulus prescaler is a logical extension of the dual-modulus prescaler. It offers four

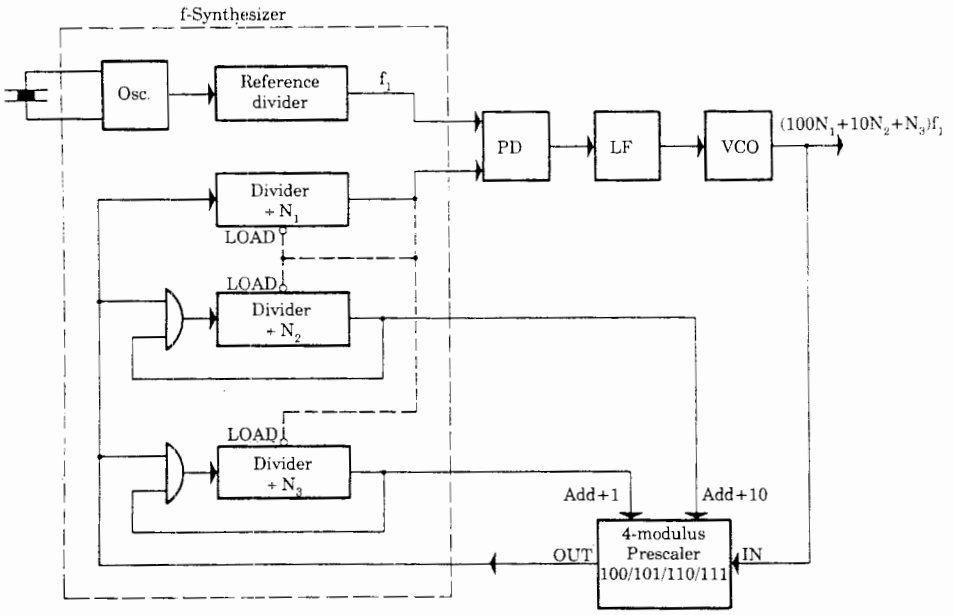


Figure 3.5 Frequency synthesizer using a four-modulus prescaler. This extends the high end of the frequency range, while allowing lower frequencies than those obtainable from a synthesizer with a dual-modulus prescaler.

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. 3.5 can divide by factors of 100, 101, 110, and 111.[11,36] 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 from Fig. 3.5 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_{tot} of this arrangement is given by

$$N_{tot} = 100N_1 + 10N_2 + N_3$$

In this equation N_3 represents the units, N_2 the tens, and N_1 the hundreds of the division ratio N_{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 ($N_{1min} = 9$).

The smallest realizable division ratio is consequently

$$N_{tot,min} = 100 \cdot 9 = 900$$

which is lower roughly by a factor of 10 than in the previous example. For a reference frequency f_1 of 10 kHz, the lowest frequency to be synthesized is therefore $900f_1 = 9$ 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_{tot} is then 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

$$N_{tot} = 2 \cdot 111 + 1 \cdot 101 + 7 \cdot 100 = 1023$$

pulses, which is exactly the number desired.

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. 3.6. The basic synthesizer circuit employed here corresponds to the simple system shown in Fig. 3.2. Of course, all synthesizer systems using dual- and four-modulus prescalers can be combined with a mixer. In the system of Fig. 3.6 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 in the *baseband* Nf_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.

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. 3.7. 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. 3.6, 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

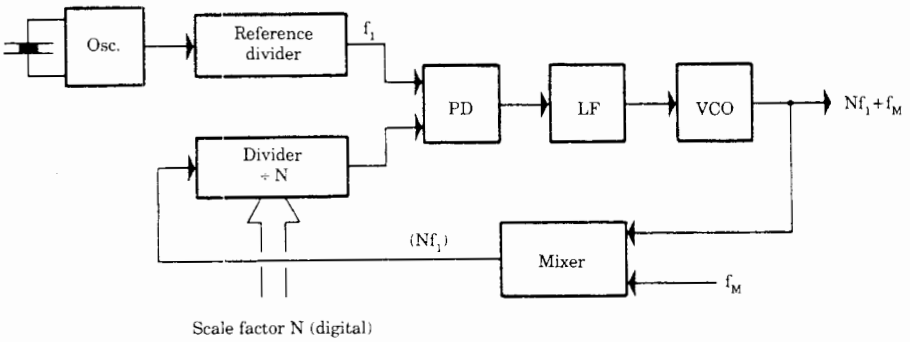


Figure 3.6 Frequency synthesizer with a mixer to extend the high end of the frequency range.

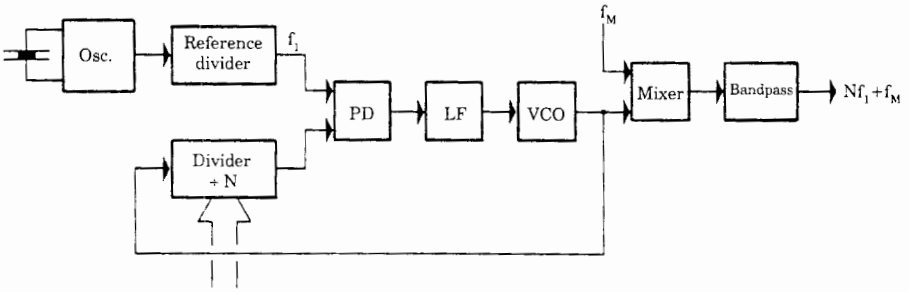


Figure 3.7 Frequency synthesizer using a mixer to extend the upper end of the frequency range. In contrast to the circuit shown in Fig. 3.6, here the mixer is outside of the loop.

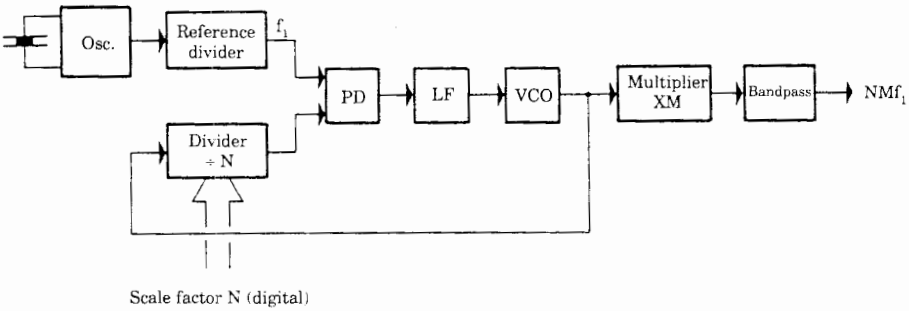


Figure 3.8 Frequency synthesizer using a frequency multiplier to extend the upper end of the frequency range.

of frequency synthesizers is given by the *frequency multiplier*, as shown by Fig. 3.8. Frequency multipliers are normally built from nonlinear elements which 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

$$f_{out} = MNf_1$$

It should be noted that now the channel spacing is not equal to the reference frequency f_1, but to Mf_1.

To illustrate the theory of frequency synthesizers, we now will design an actual system that uses the design procedure described in Sec. 2.9 and by the flowchart in Fig. 2.48.

3.2.2 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 2MHz with a channel spacing of 10kHz; i.e., frequencies of

1000, 1010, 1020, . . . , 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. 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. 2.48 (Sec. 2.9).

Step 1. Determine ranges of input and output frequencies. The input frequency is constant, $f_1 = 10\,\text{kHz}$. The output frequency is in the range from 1 to 2 MHz; thus $f_{2\min} = 1\,\text{MHz}$, $f_{2\max} = 2\,\text{MHz}$.

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 ζ. Selecting $\zeta = 0.7$ at $N = N_{\text{mean}}$ yields the following minimum and maximum values for ζ:

$$\zeta_{\min} = 0.59 \text{ for N} = 200$$
$$\zeta_{\max} = 0.83 \text{ for N} = 100$$

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 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. 3.9 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, i.e., some hertz. From this characteristic, the VCO gain becomes $K_0 = 2.24 \cdot 10^6\,\text{rad}\cdot\text{s}^{-1}\text{V}^{-1}$. Following the rules indicated in the data sheet, resistors R_1, R_2, and capacitor C_1 (Fig. 3.10) are found from graphs. The resistors must be chosen to be in the range 3 to 300 kΩ. The parallel connection of R_1 and R_2 should furthermore yield an equivalent resistance of more than 2.7 kΩ. We obtain

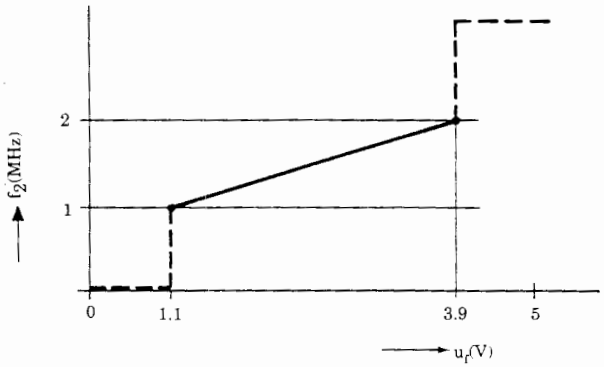


Figure 3.9 Characteristic of the VCO for the CMOS IC type 74HC/HCT4046.

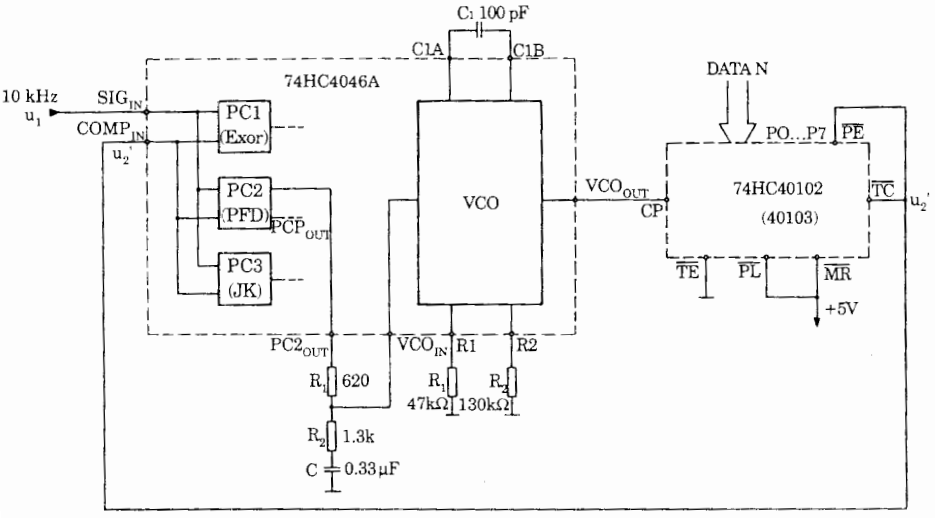


Figure 3.10 Schematic diagram of the PLL frequency synthesizer.

$$R_1 = 47\ \text{k}\Omega$$
$$R_2 = 130\ \text{k}\Omega$$
$$C = 100\ \text{pF}$$

Step 14. Loop filter selection. As indicated above, 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, e.g., within 2 ms. Hence we set $T_L = 2\,\text{ms}$. The procedure continues with step 21.

Step 21. Given T_L, the natural frequency ω_n is calculated:

$$\omega_n = \frac{2\pi}{T_L} = 3140\ \mathrm{s}^{-1}$$

The procedure continues with step 22.

Step 22. Because a passive loop filter is used, the formula for ω_n in Table 2.1 can be used only to calculate the sum $\tau_1 + \tau_2$. We obtain

$$\tau_1 + \tau_2 = 644\ \mu\mathrm{s}$$

The procedure continues with step 18.

Step 18. Given ω_n, we use the formula for ζ in Table 2.1 to calculate τ_2. We obtain

$$\tau_2 = 445\ \mu\mathrm{s}$$

Now the time constant τ_1 can be computed. Because $\tau_1 + \tau_2 = 644\,\mu\mathrm{s}$ (from step 22) and $\tau_2 = 455\,\mu\mathrm{s}$, τ_1 becomes

$$\tau_1 = 199\ \mu\mathrm{s}$$

Step 19. We calculate the loop filter components. Given τ_1 and τ_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 $C = 0.33\,\mu\mathrm{F}$ gives the resistors (rounded to the next values of the R24 series)

$$R_1 = 620\ \Omega$$
$$R_2 = 1.3\ \mathrm{k}\Omega$$

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. 3.10. The 74HC4046A PLL contains three phase detectors, PC1 (EXOR), PC2 (PFD), and PC3 (JK-flipflop). PC2 has two outputs, PC2OUT and PCPOUT. PC2OUT is the phase PFD output, and PC2OUT is an in-lock detection signal, i.e., a logical signal that becomes high when the PLL has acquired lock.

Only the PC2OUT is used in this application. For the down-scaler, an 8-bit presettable downcounter 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 ports P0 to P7 are loaded into the counter. The counter counts down on every positive transition at the CP (count pulse) input. If the counter has counted down to 0, the TC (terminal count) output 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.[52] The author repeated the design using this program and got design parameters very similar to those obtained by his own procedure.

3.2.3 Fractional-*N* frequency synthesizers

In the design example of Sec. 3.2.2 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_{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 ω_0' is much larger than the natural frequency ω_n of the PLL. Typically ω_0' is about 20 times the natural frequency. Note that in a PLL frequency synthesizer the down-scaled center frequency ω_0' is identical with $2\pi f_{\text{ref}}$; i.e., the reference period equals one period of down-scaled center frequency. Now we remember that the lock-in time is approximately equal to one period of natural frequency; i.e.,

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

from Eq. (2.82). Because the down-scaled center frequency is larger by the factor 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 (e.g., in 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—i.e., in the type of synthesizer we considered hitherto—the output frequency has always been an integer multiple of the reference frequency, e.g., $10f_{\text{ref}}$, $11f_{\text{ref}}$, $12f_{\text{ref}}$, etc. Assuming $f_{\text{ref}} =$ 10 kHz, we could create frequencies of 100, 110, 120, . . . kHz. Let us think now about a divide-by-N counter that is not only able to scale down by 10, 11, 12, but also by 10.0, 10.1, 10.2, etc. If such a down-scaler were realizable, we could create output frequencies of $10f_{\text{ref}}$, $10.1f_{\text{ref}}$, $10.2f_{\text{ref}}$, ... Still using $f_{\text{ref}} = 10$ kHz, we now would be able to create the frequencies 100, 101, 102, . . . kHz. To obtain a channel spacing of 10 kHz we now could choose $f_{\text{ref}} = 100$ kHz, which is ten times the "old" reference frequency! So doing, the natural frequency could also be increased by a factor of 10, and the loop would be 10 times as fast as the "old" one.

This is one of the ideas behind the so-called fractional-N frequency synthesizer. There are other reasons that lead to the design of fractional-N synthesizers, of course: there could be a demand to produce signals whose frequency can be any multiple of a precise frequency standard, e.g., 123.45 kHz or 146.73 kHz. Signal generators are an example for those applications.

Since a down-scaler is always a digital circuit, it can certainly not divide by fractional numbers like 10.1 or 10.2, but only either by 10 or by 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 the average this counter effectively divides the input frequency by 10.5. Dividing by 10.1, however, becomes realizable, if the counter divides by 11 during an interval whose duration equals 11 input pulses and by 10 during an interval whose duration equals 90 input pulses. During the succession of these two intervals the input of the counter received 101 pulses; the counter delivered 10 output pulses in the same period of time, so it divided by 10.1 *on average*!

Fractional division ratios of any complexity can be realized (at least theoretically). A ratio of 27.35 is obtained, for example, if a $\div N$ counter is forced alternately to divide by 28 in 35 intervals having a duration of 28 input pulses, and to divide by 27 in 65 intervals having a duration of 27 input pulses. A frequency synthesizer capable of generating output frequencies which are a fractional multiple of a reference frequency is called a *fractional N loop*.[10,39] Figure 3.11 shows the block diagram of a fractional-N loop. Its division ratio is the number $N.F$, 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. 3.11, separated by a dashed line, shows an ordinary frequency-synthesizer system generating an output signal whose frequency is N times the reference frequency f_{ref}. The block labeled "pulse-removing circuit" and the summing block Σ should be disregarded for the moment. The integer and fractional parts (N and F, respectively) of the division ratio $N.F$ are stored in a buffer register as shown in the lower part of Fig. 3.11. 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 8-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_{out} by fractional numbers. Let us see, for example, how the system can generate an output frequency $f_{out} = 5.3 f_{ref}$. We can understand the operation of the fractional N loop by examining the waveforms shown in Fig. 3.12. 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. 3.12. The signal u_2** shows 53 cycles during the time interval when the reference signal u_1 is executing 10 reference cycles. (In the following, the term *reference*

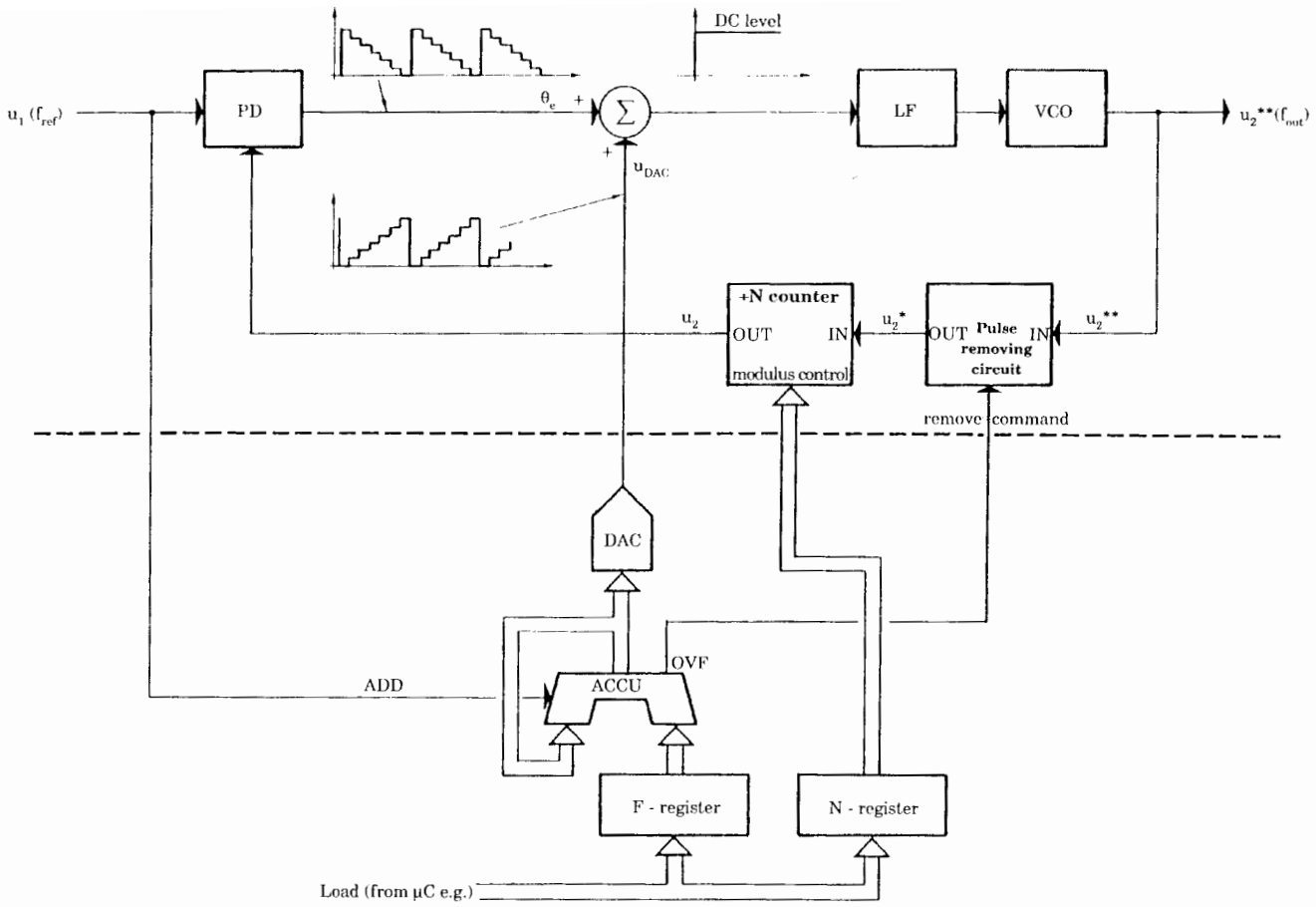


Figure 3.11 Fractional-*N* frequency synthesizer block diagram.

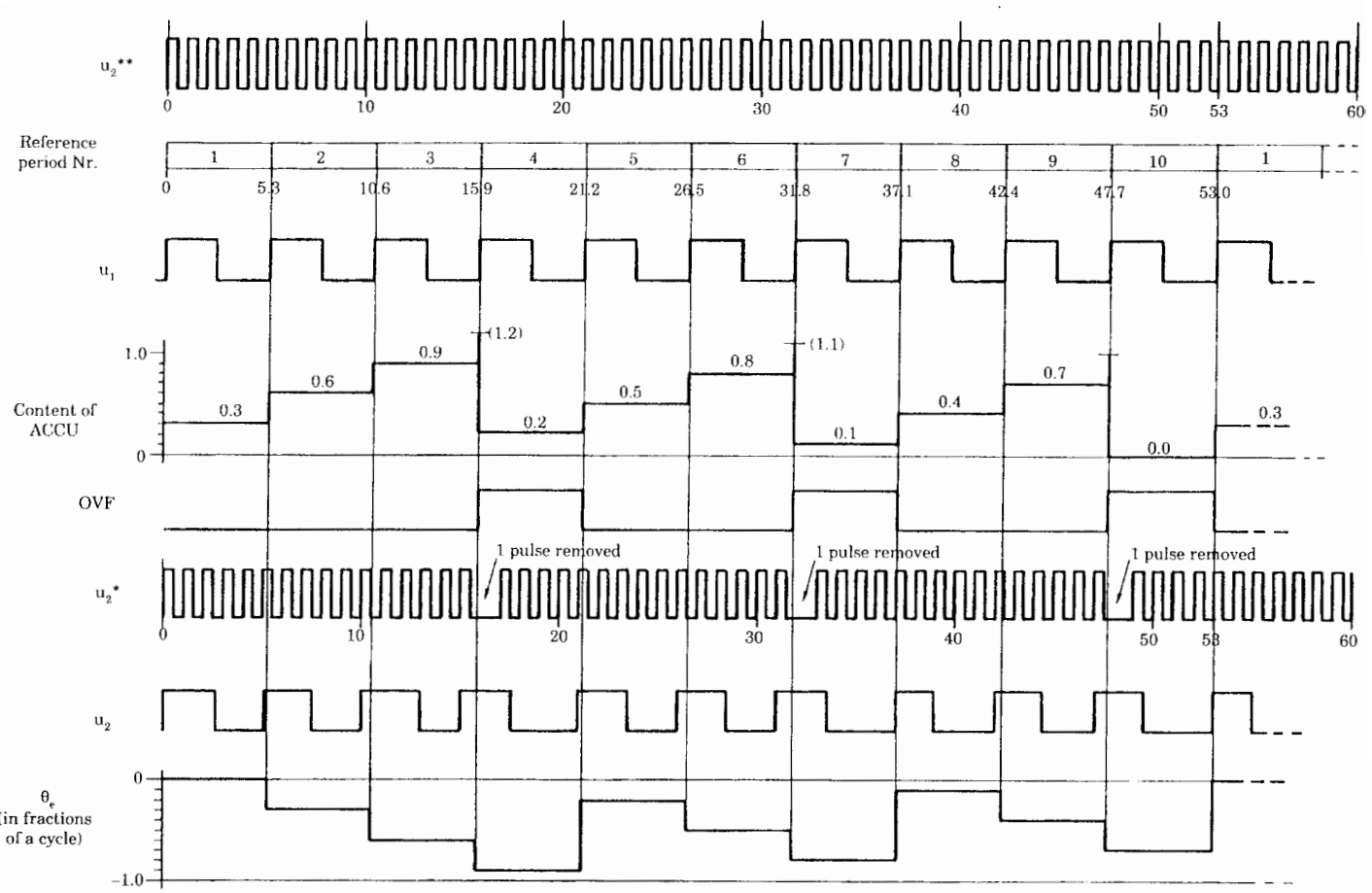


Figure 3.12 Waveforms explaining the operating principle of the fractional N loop.

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 $t = 0$.

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 pulse is then "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 from Fig. 3.12, 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 cycle during the first reference period, indicating that the synthesizer has "missed" 0.3 pulse 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 cycle in the first reference period, the total error has now accumulated to 0.6 cycle. In the third reference period the accumulated error is 0.9 cycle, 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. 3.12). The content of the ACCU is now 0.2, as seen in Fig. 3.12. 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 had divided by 6 instead of 5. As Fig. 3.12 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 10 reference periods. Because the $\div N$ counter divides by 5 forever, $10 \cdot 5 + 3 = 53$ pulses are produced by the VCO during 10 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_{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. 3.12. 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 cycle, and so on. The phase error θ_e is plotted versus time in Fig. 3.12; 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. 3.12 show that the content of the ACCU has the same amplitude as the phase error θ_e but opposite in polarity. The content of the ACCU is therefore

converted to an analog signal by a DAC (digital-to-analog converter); the output signal u_{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.

3.3 Single-Loop and Multiloop Frequency Synthesizers

In Section 3.2 we exclusively considered frequency synthesizers that were built from one single loop; such synthesizers were capable of creating a set of frequencies that were an integer (or a fractional) multiple of a given reference frequency. We will see in this section that single-loop synthesizers can become impractical if it becomes necessary to generate a large range of frequencies with a very small channel spacing.

Think, for example, of a synthesizer that would be required to generate frequencies in the range from 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 from 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; i.e. it would settle in a perhaps 20 ms. Second, it would probably have poor noise performance; as will be shown in Sec. 3.4, 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 coarse synthesizer creating the frequency range from 100 to 200 MHz in steps of 1 MHz, and a fine synthesizer generating frequencies in the range 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 (Fig. 3.13).

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 sideband with frequency $f_1 + f_2 = 100.001$ MHz and a lower sideband with fre-

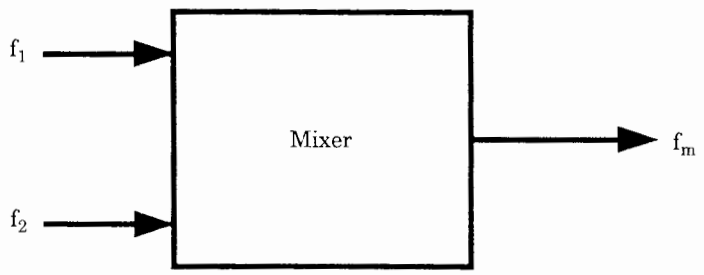


Figure 3.13 Using a mixer to add a coarse frequency f_1 and a fine frequency f_2.

quency $f_1 - f_2 = 99.999\,\text{MHz}$. Because we want to keep only 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. 3.13 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 has 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_{ofs} is the frequency offset. How large has f_{ofs} to be chosen in order to safely separate upper and lower sidebands at the mixer output?

Let the minimum coarse frequency be $f_{1\text{min}}$ and the maximum coarse frequency be $f_{1\text{max}}$. In analogy let the minimum fine frequency be $f_{2\text{min}}$ and the maximum $f_{2\text{max}}$. Then the maximum frequency in the upper sideband would be

$$f_{u\text{max}} = f_{1\text{max}} + f_{\text{ofs}} + f_{2\text{max}}$$

and the minimum frequency in the upper sideband

$$f_{u\text{min}} = f_{1\text{min}} + f_{\text{ofs}} + f_{2\text{min}}$$

For the lower sideband the maximum and minimum frequencies would be

$$f_{l\text{max}} = f_{1\text{max}} - f_{\text{ofs}} - f_{2\text{min}}$$
$$f_{l\text{min}} = f_{1\text{min}} - f_{\text{ofs}} - f_{2\text{max}}$$

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

$$f_{u\text{min}} > f_{l\text{max}}$$

Assuming $f_{2\text{min}} = 0$, this inequality leads to

$$f_{\text{ofs}} > \frac{f_{1\text{max}} - f_{1\text{min}}}{2}$$

In our example f_{ofs} would have to chosen larger than 50 MHz. Figure 3.14 is a possible implementation of the intended synthesizer.

The circuit consists of two synthesizers; hence is a dual-loop system. The frequency offset has been chosen $f_{\text{ofs}} = 75\,\text{MHz}$ here. Let us consider the upper (coarse) synthesizer first. Without mixer Mix1 this loop would create frequencies in the range 25 to 125 MHz with a channel spacing of 1 MHz. Assume for the moment that $f_2 = 0$; i.e., the frequency at the output of bandpass filter BP2 is exactly 75 MHz. Because of the mixer, VCO1 is forced now to create an output

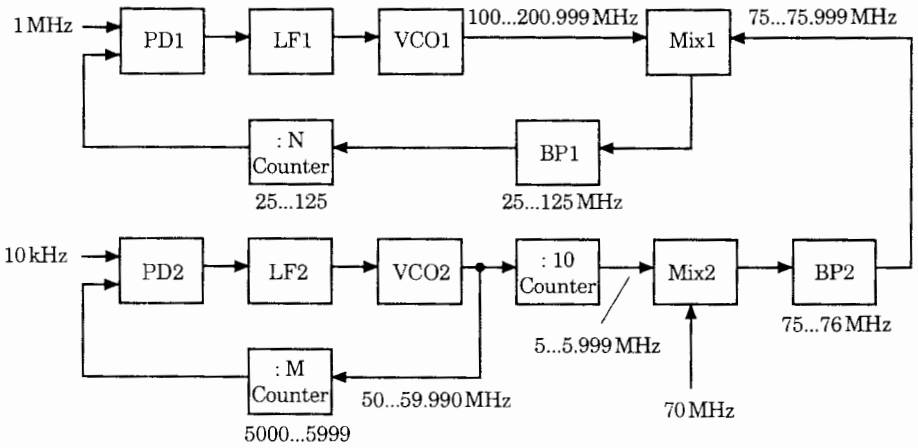


Figure 3.14 Dual-loop frequency synthesizer built from a synthesizer creating coarse frequency steps (1 MHz, upper part) and from another synthesizer creating fine frequency steps (1 kHz, lower part).

frequency that is higher by that offset; i.e., the output frequency range of VCO1 is now 100 to 200 MHz. Bandpass filtering 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 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 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 channel spacing 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 5 to 5.999 MHz with channel 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 and 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.

Multiloop synthesizers are frequently found in signal generators, receivers, transmitters (e.g., short wave), and the like. Very sophisticated multiloop designs have been described by Rohde.[48,49]

3.4 Noise in 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 some phase jitter, and moreover, we can detect a couple of sidebands (so-called spurs) around the desired center frequency. In this section 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 today that greatly simplify the analysis. Basically, each part of the synthesizer circuit can contribute to output phase jitter.[48] In the following we will concentrate on the dominant ones. Figure 3.15 shows a simplified model for the determination of output phase jitter and spurious sidebands. Three sources of phase jitter and spurs can be recognized:

1. Phase jitter created by the reference oscillator. Even the highest-quality reference oscillator is not free from output phase jitter. This perturbation is denoted $\theta_{n,\mathrm{ref}}$ in Fig. 3.15.
2. 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 $\theta_{n,\mathrm{VCO}}$ in Fig. 3.15
3. As we have seen in Chap. 2.3.1, the phase detector delivers an output signal that is proportional to the phase error θ_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 case of the PFD there can even be an ac component that 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-

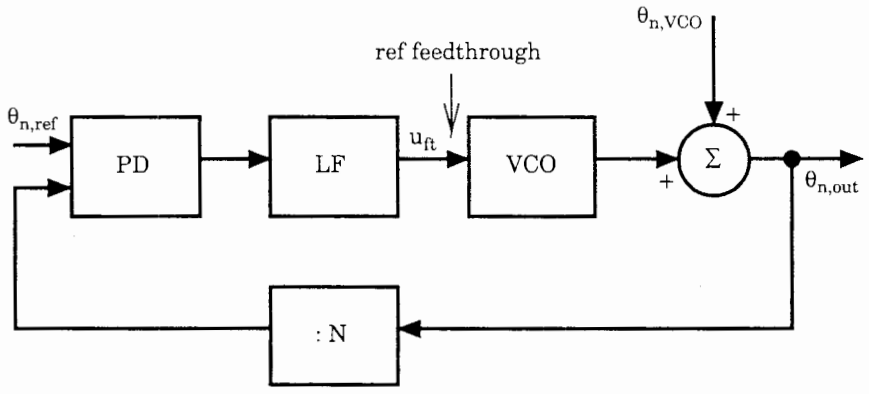


Figure 3.15 Model for analysis of output phase jitter $\theta_{n,\mathrm{out}}$ in a PLL frequency synthesizer.

modulate the VCO output. When the frequency of that ac signal is equal to the reference frequency f_{ref}, for example, we will observe "spurs" at a distance of $\pm f_{ref}$, $\pm 2f_{ref}$, . . . from the carrier frequency f_0. The signal causing those spurs is denoted u_{ft} (ft = feedthrough) in Fig. 3.15.

All these noise sources contribute to output phase jitter (labeled $\theta_{n,out}$ in Fig. 3.15) and spurs. 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 are therefore going to analyze the three mentioned effects.

3.4.1 Phase jitter $\theta_{n,ref}$ of the reference oscillator

To analyze phase noise at the output of the reference oscillator, let us recall the noise theory presented in Sec. 2.8.4. Fig. 2.44 has shown the relationships between signal power P_s, noise power P_n, and input phase jitter θ_{n1}. In Fig. 2.44*a* the power spectral density (PSD) of noise power P_n was shown. This power spectrum has one-sided bandwidth $B_i/2$ and is symmetrical around the center frequency f_0 of the PLL. Fig. 2.44*b* represented the PSD of input phase jitter θ_{n1}. This spectrum also has the one-sided bandwidth $B_i/2$ 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

$$u_1(t) = U_{10} \sin(\omega_1 t + \theta_{n1}(t)) \tag{3.4}$$

[see Eq. (2.2)]. 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. 2.44*a*), 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. Eq. (2.112) gave the relationship

$$\overline{\theta_{n1}^2} = \frac{P_n}{2P_s} \quad [\text{rad}^2] \tag{3.5}$$

$\overline{\theta_{n1}^2}$ 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. 2.44*b*. 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. 2.44*b*, $\overline{\theta_{n1}^2}$ 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*, i.e.

$$\overline{\theta_{n1}^2} = \frac{P_n}{P_s} \qquad \text{rad}^2 \tag{3.6}$$

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 (3.6) tells us how large the phase jitter will be when signal power P_s and noise power P_n are given. All variables in Eq. (3.6) represent power quantities. This is very clear for the variables P_n and P_s, but the variable $\overline{\theta_{n1}^2}$ can be considered as power as well, because it is the mean square value of input phase jitter $\theta_{n1}(t)$. When analyzing noise performance in Sec. 2.8.4, we started from the premise that the noise spectrum would be "white"; i.e., 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. (3.6), which yields

$$S_{\theta\theta}(f_m) = \frac{S_{nn}(f_m)}{P_s} \qquad \text{rad}^2/\text{Hz} \tag{3.7}$$

where $S_{\theta\theta}(f_m)$ is the power spectral density of phase perturbation (jitter) θ_{n1} at (modulating) frequency f_m; the unit is rad^2/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 examining the impact of phase jitter onto a carrier with power $P_s = 1\,\text{mW}$ and frequency f_0. For this analysis the model in Fig. 3.16 is used. The noise source is assumed to be thermal noise. Because the phase jitter modulates the phase of the carrier, a phase modulator is shown at the left of the circuit.

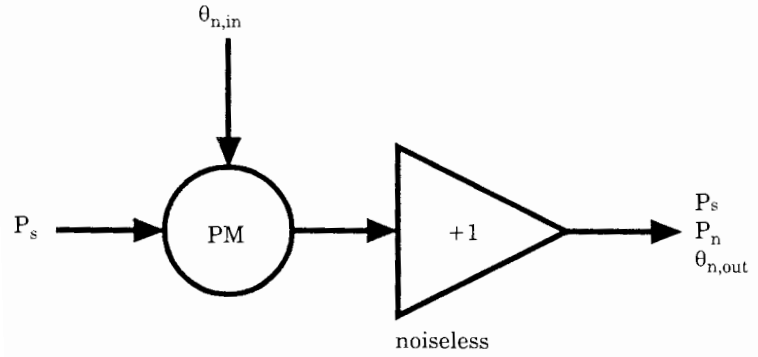


Figure 3.16 Model demonstrating the effect of phase jitter onto a carrier signal having power P_s. The phase jitter creates phase modulation of the carrier (PM = phase modulator).

One of its input signals is the carrier (P_s); the other input is thermal noise. The phase modulator is followed by an amplifier, which is assumed to be noise free. (Later we will release this premise.) The power gain of the amplifier is assumed to be 1. Because the amplifier does not add further noise, the PSD of phase jitter at the output of the amplifier $S_{\theta\theta,\text{out}}(f_m)$ is identical with the PSD of phase jitter at the input $S_{\theta\theta,\text{in}}(f_m)$. The PSD of thermal noise is defined by

$$S_{nn}(f_m) = kT \qquad \text{W/Hz} \tag{3.8}$$

with k = Boltzmann constant = $1.4 \cdot 10^{-23}$ W · s/K and T = absolute temperature in kelvins (K). 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

$$S_{\theta\theta,\,\text{out}}(f_m) = \frac{kT}{P_s} = 0.41 \cdot 10^{-17} \qquad \text{rad}^2/\text{Hz}$$

Because this is an extremely small quantity, $S_{\theta\theta,\text{out}}(f_m)$ is mostly expressed on a logarithmic scale, i.e., in decibels. We therefore introduce a new variable $S_{\theta\theta,\text{out}}(f_m)_{\text{dB}}$:

$$S_{\theta\theta,\text{out}}(f_m)_{\text{dB}} = 10\log_{10} S_{\theta\theta,\text{out}}(f_m) \qquad \text{dBc/Hz} \tag{3.9}$$

The unit of $S_{\theta\theta,\text{out}}(f_m)_{\text{dB}}$ is dBc/Hz, and for the current example, the result is $S_{\theta\theta,\text{out}}(f_m)_{\text{dB}} = -174$ 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 $S_{\theta\theta,\text{out}}(f_m)_{\text{dB}}$ stands for noise power referred to carrier power. The value $S_{\theta\theta,\text{out}}(f_m)_{\text{dB}} = -174$ dBc/Hz is the absolutely 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, i.e., the square root of $S_{\theta\theta,\text{out}}(f_m)$. This yields

$$\theta_{n,\text{rms}} = \sqrt{S_{\theta\theta,\text{out}}(f_m)} = \sqrt{0.41 \cdot 10^{-17}} = 2.02 \cdot 10^{-9} \text{ rad/Hz}$$

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 following reasons. As we know, $S_{\theta\theta}(f_m)$ as defined in Eq. (3.7) represents a ratio of noise power to carrier power. If total noise power is P_n and carrier power is P_s, a noise to carrier ratio (NCR) could be defined by

$$\text{NCR} = \frac{P_n}{P_s} \qquad \text{W/W}$$

Note that the unit of NCR, W/W, is dimensionless. Usually one defines another noise to carrier ratio NCR_{dB} expressed in decibels:

$$\text{NCR}_{\text{dB}} = 10\log_{10}(\text{NCR}) = 10\log_{10}\frac{P_n}{P_s}$$

Checking Eq. (3.7) 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 Hz^{-1}. Mathematically it is not correct to build the logarithm of a ratio that is not dimensionless. [Try to find out what $\log(Hz^{-1})$ is.] We can circumvent that dilemma by introducing new variables $P_{nn}^*(f_m)$ and $S_{\theta\theta}^*(f_m)$ as follows:

$$S_{nn}^*(f_m) = S_{nn}(f_m) \cdot B \qquad \text{W}$$
$$S_{\theta\theta}^*(f_m) = S_{\theta\theta}(f_m) \cdot B \qquad \text{rad}^2$$

B stands for bandwidth and is set $B = 1\,\text{Hz}$. As a result of 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 $B = 1\,\text{Hz}$, located at an offset f_m from the carrier frequency. The new variable $S_{\theta\theta}^*(f_m)$ 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 rad^2 and not rad^2/Hz. $S_{\theta\theta}^*(f_m)$ therefore represents no longer 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 $S_{\theta\theta}^*(f_m)$ 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 us make a numerical example. Assume that the carrier power is $P_s = 1\,\text{mW}$, and that the noise power density $S_{nn}(f_m)$ is $10^{-15}\,\text{W/Hz}$ at an offset $f_m = 10\,\text{kHz}$ from the carrier frequency. The noise power $S_{nn}^*(f_m)$ within a bandwidth of 1 Hz at $f_m = 10\,\text{kHz}$ is then given by

$$S_{nn}^*(10{,}000)_{\text{dB}} = 10^{-15}\,\frac{\text{W}}{\text{Hz}} \cdot 1\,\text{Hz} = 10^{-15}\,\text{W}$$

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

$$S_{\theta\theta}^*(f_m) = \frac{S_{nn}^*(f_m)}{P_s} = \frac{10^{-15}\,\text{W}}{10^{-3}\,\text{W}} = 10^{-12}$$

Hence $S_{\theta\theta}^*(f_m)_{\text{dB}}$ becomes

$$S_{\theta\theta}^*(f_m)_{\text{dB}} = 10\log_{10} 10^{-12} = 120\,\text{dBc}$$

We conclude that the noise power within a bandwidth of 1 Hz at offset frequency $f_m = 10\,\text{kHz}$ 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 $f_m = 10\,\text{kHz}$ is $10^{-12}\,\text{rad}^2$. For the rms value of phase perturbation in that frequency interval we get

$$\theta_{n1,\text{rms}} = \sqrt{\overline{\vartheta_{n1}^2}} = \sqrt{S_{\theta\theta}{}^*(f_m)} = \sqrt{10^{-12}} = 10^{-6} \quad \text{rad}$$

To avoid confusion with the standards used in practically all books and papers on frequency synthesizers, we will use nevertheless the familiar unit dBc/Hz for power density of phase jitter, although this unit has been shown to be somewhat incorrect.

As mentioned, real amplifiers add further noise. Noise performance of amplifiers is specified by noise figure F, which is defined by

$$F = \frac{P_{s,\text{out}}/P_{n,\text{out}}}{P_{s,\text{in}}/P_{n,\text{in}}} \tag{3.10}$$

that is, 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 $S_{\theta\theta,\text{out}}(f_m)$ by factor F; i.e., we now have

$$S_{\theta\theta,\text{out}} f(m) = F \cdot S_{\theta\theta,\text{in}}(f_m) = \frac{kTF}{P_s}$$

When we assume $F = 6\,\text{dB}$, $S_{\theta\theta,\text{out}}(f_m)_{\text{dB}}$ increases by 6 dB:

$$S_{\theta\theta,\text{out}}(f_m) = -174 + 6 = -168\ \text{dBc/Hz}$$

A much more disturbing effect in oscillators, however, is *flicker noise*. For a noiseless amplifier the only source of input noise has been thermal noise (kT). For each real amplifier, however, the equivalent input noise power increases with f_m^{-1} at frequencies below a corner frequency denoted f_c (see Fig. 3.17). For operational amplifiers, for example, f_c can be in the order of 10 Hz to some kHz. In oscillators f_c can easily extend into the MHz region.

When flicker noise is present, we get for the PSD of the noise signal

$$S_{nn}(f_m) = kTF\left(1 + \frac{f_c}{f_m}\right) \qquad \text{W/Hz} \tag{3.11}$$

Now we get for the PSD of phase jitter the expression

$$S_{\theta\theta,\text{out}}(f_m) = \frac{kTF}{P_s}\left(1 + \frac{f_c}{f_m}\right) \qquad \text{rad}^2/\text{Hz} \tag{3.12}$$

Still assuming $F = 6\,\text{dB}$, we get for a modulating frequency $f_m = f_c/10$ the value

$$S_{\theta\theta,\text{out}}(f_m)_{\text{dB}} = -174 + 6 + 10 = -158\ \text{dBc/Hz}$$

At a modulating frequency $f_m = f_c/100$ this would increase to −148 dBc/Hz.

We are ready now to compute the phase jitter at the output of an oscillator. The corresponding model is shown in Fig. 3.18.

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

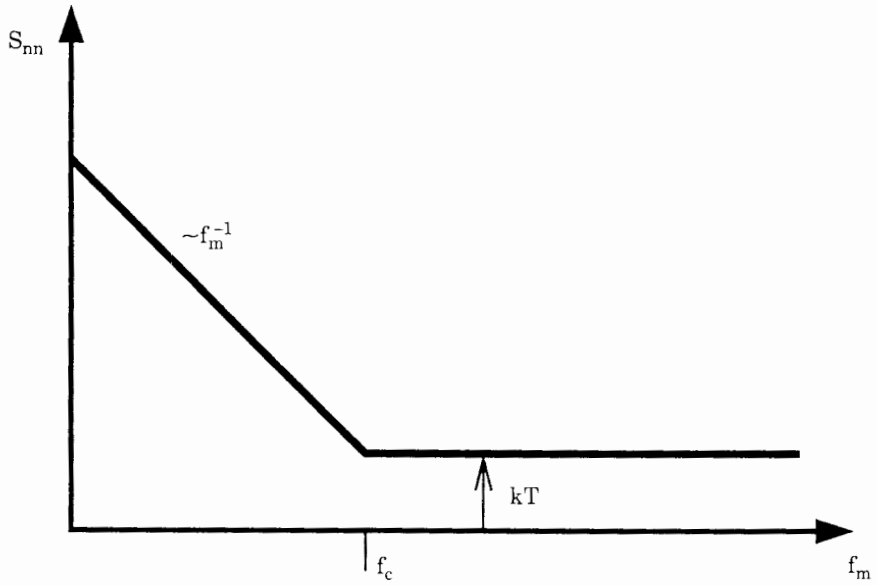


Figure 3.17 Power spectral density of input phase jitter $\theta_{n,in}$. Noise below corner frequency f_c is called *flicker noise*. Noise above f_c is *thermal noise*.

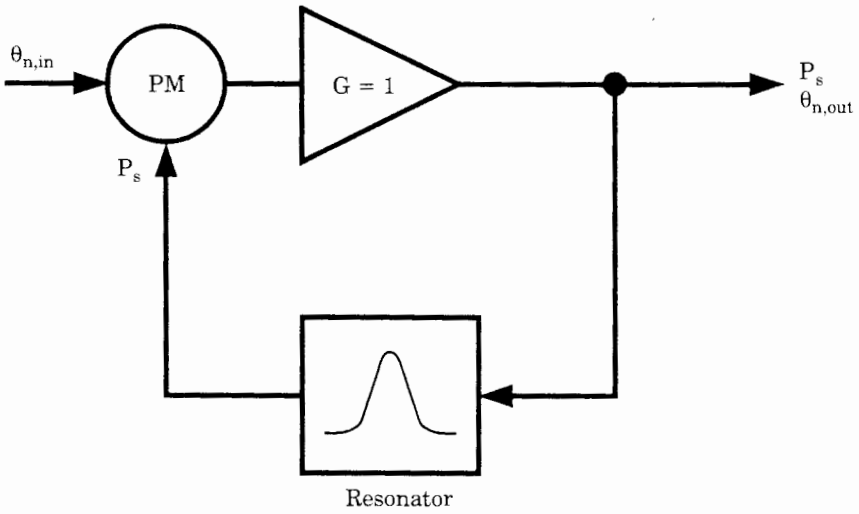


Figure 3.18 Model for the analysis of phase jitter in an oscillator. PM = phase modulator.

in the feedback path. As a result of the feedback path, $S_{\theta\theta,\text{out}}(f_m)$ no longer is identical with $S_{\theta\theta,\text{in}}(f_m)$ as in the model of Fig. 3.16, but is given by

$$S_{\theta\theta,\text{out}}(f_m) = S_{\theta\theta,\text{in}}(f_m) \cdot |G_n(f_m)|^2 \qquad \text{rad}^2/\text{Hz} \tag{3.13}$$

Here $G_n(f_m)$ is the closed-loop gain of the oscillator. As shown by Rohde,[48] the squared closed-loop gain is given by

$$|G_n(f)|^2 = 1 + \frac{1}{f_m^2}\left(\frac{f_0}{2Q}\right)^2 \tag{3.14}$$

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. 3.19*b* for two cases, a low Q ($Q = Q_2$) and a high Q ($Q = Q_1$). Figure 3.19*a* sketches once more the power density spectrum of the equivalent input noise. In addition to thermal noise kT, there is flicker noise below corner frequency f_c. The 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 us discuss first the high-Q case. The power density spectrum of the ouput phase jitter is obtained by multiplying the appropriate curves in Fig. 3.19*a* and *b*. The result is plotted in Fig. 3.19*c*. For high Q, $S_{\theta\theta,\text{out}}(f_m)$ is constant versus frequency for $f > f_c$. Under ideal conditions, $S_{\theta\theta,\text{out}}(f_m)_{\text{dB}}$ approaches −174 dB for a signal power of 1 mW (0 dBm). Below f_c, $S_{\theta\theta,\text{out}}(f_m)$ increases with f_m^{-1}, and below corner frequency $f_0/2Q_1$, it increases with f_m^{-3}.

In the low-Q case, $S_{\theta\theta,\text{out}}$ 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 dramatically from Fig. 3.19 is that 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. 3.15, 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. 2.4.2 how the PLL reacts onto phase signals $\theta_1(t)$ applied to the reference input. To compute the output phase signal $\theta_2'(t)$ we found

$$H(s) = \frac{\Theta_2'(s)}{\Theta_1(s)} \tag{3.15}$$

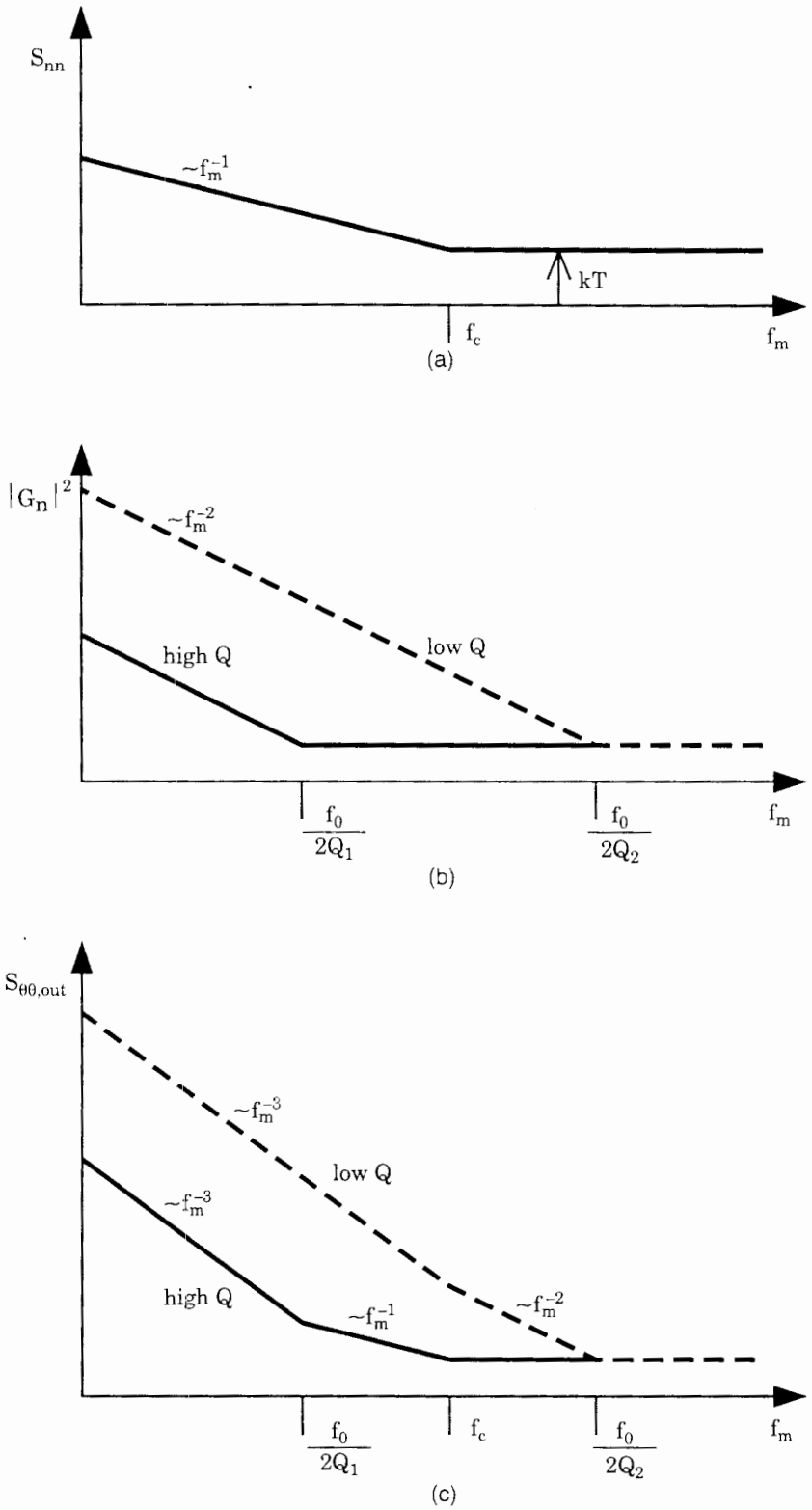


Figure 3.19 Power density spectra of phase jitter in an oscillator. (*a*) Power density spectrum of the noise applied to the oscillator input. (*b*) Squared closed-loop gain $|G_n(f_m)|^2$ of the oscillator. Solid line represents an oscillator having a resonator with high Q; dashed line represents an oscillator with a low-Q resonator ($Q_1 > Q_2$). (*c*) Power density spectrum of the oscillator's output phase jitter $S_{\theta\theta,out}(f_m)$.

[see Eq. (2.29)]. In 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 easlily seen from the block diagram in Fig. 2.1, θ_2 and θ_2' are related by $\theta_2 = \theta_2' N$. For the power spectral density of the phase jitter at the output of the VCO we consequently have

$$S_{\theta\theta,\text{out}}(f_m) = |H(f_m)|^2 \cdot N^2 \cdot S_{\theta\theta,\text{ref}}(f_m) \qquad \text{rad}^2/\text{Hz} \tag{3.16}$$

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 $f_{3\text{dB}}$), $H(f_m) \approx 1$; 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. 4.

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 in Fig. 3.1, for example. 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

$$S_{\theta\theta,\text{out}}(f_m) = |H(f_m)|^2 \cdot \frac{N^2}{R^2} \cdot S_{\theta\theta,\text{ref}}(f_m) \qquad \text{rad}^2/\text{Hz} \tag{3.17}$$

We note that the phase frequency response $H(f)$ acts as a low-pass filter onto the phase jitter caused by the reference oscillator. To mimimize 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 ω_n, i.e., a large lock-in time.

3.4.2 Phase jitter $\theta_{n,\text{VCO}}$ of the VCO

In Sec. 3.4.1 we investigated the phase jitter introduced by the reference oscillator. Checking the phase jitter model of Fig. 3.15 once again, we recognize that the VCO is nothing more than an oscillator, too; hence it will introduce additional phase jitter into the loop. This phase noise (denoted $\theta_{n,\text{VCO}}$ in the block diagram) enters the loop at the input of summator labeled Σ. If there were no feedback, phase noise $\theta_{n,\text{VCO}}$ would simply add to the ouput phase noise $\theta_{n,\text{out}}$. Because we have a feedback path, we must compute the closed-loop gain G_n from the insertion point of the VCO phase noise $\theta_{n,\text{VCO}}$ to the output terminal $\theta_{n,\text{out}}$. For G_n we obtain the very simple result

$$G_n(f) = H_e(f) \tag{3.18}$$

i.e., is identical with the *error frequency response* as defined in Eq. (2.36). As can be seen from the Bode diagram for $H_e(\omega)$ in Fig. 2.23, $H_e(\omega)$ is a *high-pass* function. The PSD of phase noise contributed by the VCO is given by

$$S_{\theta\theta,\text{out}}(f_m) = |H_e(f_m)|^2 \cdot S_{\theta\theta,\text{VCO}}(f_m) \qquad \text{rad}^2/\text{Hz} \tag{3.19}$$

with $S_{\theta\theta,\text{VCO}}(f_m)$ = PSD of phase jitter introduced by the VCO, $S_{\theta\theta,\text{out}}(f_m)$ = PSD of output phase jitter due to VCO phase jitter, and f_m = frequency offset from center frequency f_0. The bandwidth of high-pass function $H_e(f)$ is the same as the bandwidth of the low-pass function $H(f)$, because $H_e(s) = 1 - H(f)$, from Eq. (2.37). 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. 3.4.1 where we stated that the loop bandwidth should be made small in order to reduce reference phase jitter.

The theory on VCO phase jitter has another weak point. Although a VCO is an oscillator, its output phase jitter cannot be analyzed by the model used for the reference oscillator, Fig. 3.18. Whereas the reference oscillator could be modeled by an amplifier in the forward path and a resonator in the feedback, this is not valid for VCOs because the frequency created by a VCO is not determined by a resonant circuit but by the time interval required to charge a capacitor to some threshold level. Little is known about phase jitter created by VCOs; what we know is that most of the noise power is concentrated near the oscillating frequency and hence can be considered as a kind of flicker noise.

3.4.3 Reference feedthrough 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, i.e., 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 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 Nf_{ref}, and the ripple signal has a fundamental frequency of $2f_{ref}$. For $N = 3$, spurious sidebands appear at the VCO output whose frequencies are $Nf_{ref} \pm 2f_{ref}$, $Nf_{ref} \pm 4f_{ref}$, etc. 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 phase detector. In the locked state the JK-flipflop generates an output signal whose frequency is identical with the reference frequency (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 $Nf_{ref} \pm f_{ref}$, $Nf_{ref} \pm 2f_{ref}$, etc.

When the PFD is used as phase detector, its output signal is theoretically in the 0 state, when the VCO operates exactly on its center frequency, as shown in Fig. 2.13*a*. Under normal PLL operation, however, the PFD will output correction pulses, as shown in Fig. 2.13*b* and *c*. The polarity of the correction pulses will change periodically, so positive and negative correction pulses will appear in succession. Fortunately the duration of the correction pulses is short when the PLL is locked, so the ripple on the VCO input signal will be much smaller than in the two former cases. It is of vital interest, of course, to know the power of such sidebands. 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:

$$S_1 = \frac{P_s}{P_{sb1}} \tag{3.20}$$

where P_s is the power of the synthesized signal and P_{sb1} is the power of the first spurious sideband. Mostly, a logarithmic quantity $S_{1(dB)}$ is specified:

$$S_{1(dB)} = 10 \log S_1 \tag{3.21}$$

If the EXOR or the JK-flipflop is used, an approximation for $S_{1(dB)}$ is given by[15]

$$S_{1(dB)} = 20 \log \frac{K_0 U_B}{2\pi^2 f_r} |F(2\pi f_r)| \tag{3.22}$$

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 another approximation has been found[15]:

$$S_{1(\mathrm{dB})} = 20\log(K_0 U_B \tau)|F(2\pi f_r)| \tag{3.23}$$

Here the ripple frequency f_r is not necessarily identical with the reference frequency but can be considerably lower, as will be discussed in the following. τ is the width of the correction pulses. The following consideration yields an approximation for the quantity τ. When a PLL using the PFD as phase detector 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. 3.20*a*) 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.

Figure 3.20*b* shows the PFD output signal u_d that 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, which is denoted τ_1 here. The width of the

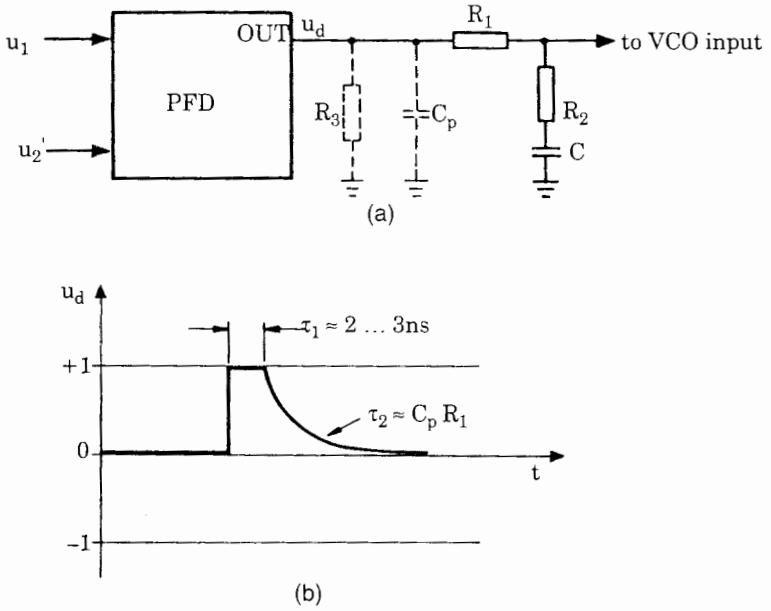


Figure 3.20 The figure explains the backlash effect of the PFD and the effect of parasitic capacitance at the phase detector output. (*a*) Schematic of the PFD including parasitic capacitor C_p. The loop filter is also shown. (*b*) Waveform of the u_d signal. The duration τ_1 of the output pulse cannot be less than the backlash interval.

correction pulse is nearly equal to τ_1 in this case. Unfortunately, each real device contains parasitic capacitances. In our example, parasitic capacitance C_p (Fig. 3.20*a*) 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 Ω. Thus τ_2 will also be in the order of 5 ns or more. This means that the value of τ in Eq. (3.26) is approximately given by the sum $\tau_1 + \tau_2$ and is typically around 10 ns. Because the correction pulses cannot be arbitrarily narrow, the pulse as depicted in Fig. 3.20*b* alters the instantaneous frequency of the VCO more than would normally be required. In the example of Fig. 3.20*b* 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 produce a correction pulse (of negative polarity), however, only 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.) This undesired subharmonic frequency modulation can be reduced in different ways.

One way would be to use a higher-order loop filter. For a first-order loop filter the gain at higher frequencies does not decay towards zero (see the magnitude response in Fig. 2.17) but stays constant, typically in the range 0.1 to 0.3. Choosing the cutoff frequency of the filter very much lower than the reference frequency does not bring any benefit because the filter gain does not roll off toward zero. With a higher-order filter, however, the gain will roll off toward zero; for a second-order filter, the slope of the magnitude response will be –20 dB/decade, with a third-order filter the slope will be –40 dB/decade, etc. (Note that the slope of a second-order filter without a zero would be –40 dB/decade; because we need a zero for reasons of stability, that slope is reduced to –20 dB/decade.) In Chap. 4 we will deal with higher-order filters in more detail.

There is still another way to eliminate subharmonic frequency modulation. This is sketched in Fig. 3.20*a* by the dashed resistor R_3. Normally, R_3 is a high-value resistor that slowly but steadily discharges filter 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 $Nf_{ref} \pm f_{ref}$, $Nf_{ref} \pm 2f_{ref}$, etc.

Another way to force the PFD to output correction pulses in every reference cycle has been realized in the PFD that is incorporated in the integrated circuit type 74HCT9046A manufactured by Philips (see Sec. 2.7 and Table 10.1). The data sheet of the 74HCT9046A can be downloaded from *http://www.philipslogic.com/products/plls/9046/*.[51] This IC contains two phase detectors, an EXOR and a PFD. The latter has a charge pump output as explained in Sec. 2.7. With reference to Fig. 2.40*c*, the timing of both current sources is modified such that the outputs of the upper and of the lower current source overlap during an interval of approximately 15 ns; i.e., both current sources are turned on during at least 15 ns in every reference cycle. The timing of the current outputs is demonstrated by Fig. 3.21 for three cases:

1. The signal u_1 leads the signal u_2'.
2. The signal u_2' leads the signal u_1.
3. 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 that 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 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

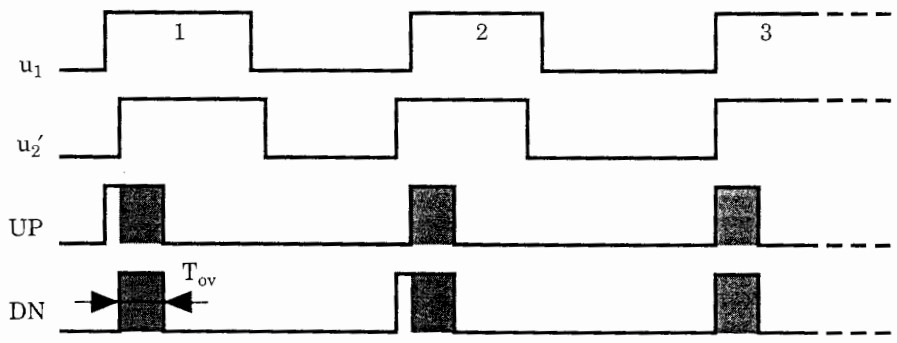


Figure 3.21 Pulse timing of the PFD in the 74HCT9046A circuit. The shaded areas characterize the overlapping interval (T_{ov}), where $T_{ov} \approx 15$ ns.

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 case of the 74HCT9046A, however, the upper current source in Fig. 2.40*c* turns on on the positive edge of u_1; 1 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 during 15 ns; hence the lower would conduct current for 16 ns, the upper for 15 ns, which results 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. 3.20) 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.

Chapter

4

Higher-Order Loops

4.1 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 Fig. 2.16. As can be seen from the Bode diagrams of the loop filter transfer functions in Fig. 2.17 the gain rolls off at −20 dB/dec at higher frequencies but asymptotically approaches a nonzero value for radian frequencies higher than $1/\tau_2$, where τ_2 is the time constant in the numerator of the filter transfer function (see Eqs. 2.26 to 2.28). In the discussion of spectral purity of PLL frequency synthesizers, we recognized that reference frequency feedthrough becomes a problem. Spurious sidebands 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, i.e., 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° at higher frequencies, and a zero creates a phase shift of +90°. When the poles and the zeros are properly located the overall phase shift never comes close to −180°; hence the loop stays stable. This goal was easily met by choosing time constant τ_2 of the loop filter such that a reasonable damping factor ζ was obtained. If the loop filter has two or more poles, the phase shift can become larger than 180°; hence the poles and zeros of the loop filter must be placed such that stability is maintained. We will deal with stability of loops in the following section.

4.2 Analyzing 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, i.e., 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

$$F(s) = \frac{1 + sT_2}{(1 + sT_1)(1 + sT_3)} \tag{4.1}$$

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

$$G(s) = \frac{K_0 K_d}{N} \frac{(1 + sT_2)}{s(1 + sT_1)(1 + sT_3)} \tag{4.2}$$

(also refer to the PLL model in Fig. 2.21).

To analyze stability, the Bode diagram for $G(s)$ is plotted. This is done by replacing s by $j\omega$ and plotting magnitude and phase of $G(\omega)$ versus radian frequency ω. The magnitude plot is logarithmic on both axes; the phase plot is logarithmic on the frequency axis, but linear on the phase axis. The Bode plot is shown in Fig. 4.1. In the magnitude plot, the exact (dashed curve) and the asymptotic diagram (solid curve) are shown. The asymptotic curve consists of line segments and is an approximation to the exact magnitude function. The

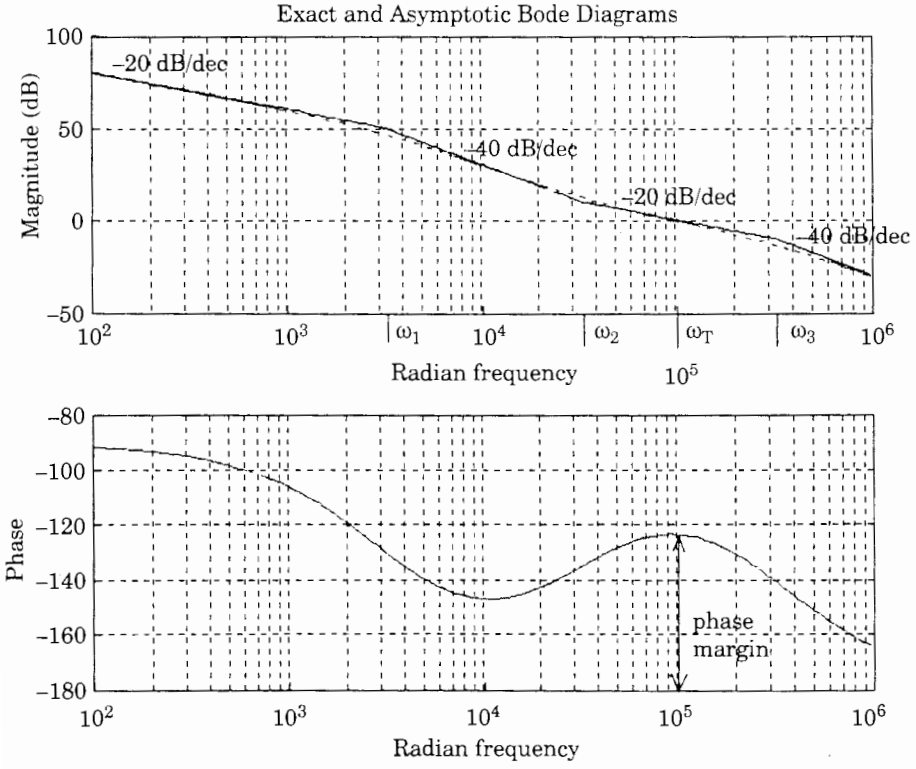


Figure 4.1 Analyzing loop stability by the Bode diagram.

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 ω_1 and ω_3 are created by the poles of the transfer function, the corner at ω_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/dec (dec = decade of frequency). Above $\omega = \omega_1$ the first pole of the loop filter comes into play; now the asymtpotic value of slope is increased to –40 dB/dec. 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/dec again. Above $\omega = \omega_3$, however, the slope becomes –40 dB/dec because of the second pole of the loop filter.

Now let us look at the phase plot. For minimum phase networks, phase plot and magnitude plot are closely related. (A system is minimum phase if all the poles and zeros 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° when the slope of the magnitude plot is –20 dB/dec. The phase plot asymptotically approaches –180° if the magnitude slope is –40 dB/dec, etc. We see from the phase plot that the curve starts at –90° at low frequencies. Above $\omega = \omega_1$ the phase bends down towards –180°. Above $\omega = \omega_2$, however, the asymptotic value of phase becomes –90° because of the onset of the filters' zero. Finally, above $\omega = \omega_3$ the asymptotic value of the phase becomes –180° again. We note that the phase curve exhibits a "peak" between frequencies ω_2 and ω_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* ω_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°*. Let us denote that phase value by $\varphi(\omega_T)$. The quantity $180° + \varphi(\omega_T)$ is called *phase margin* φ_m. In control engineering, one attempts to get phase margins between about 30 to 60°. 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. 4.1 the phase margin is about 55°. It is clear that the phase in the range $\omega_2 < \omega < \omega_3$ would closely approach –90° when this region would be chosen larger, i.e., would span some decades of frequency. The closer the two frequencies ω_2 and ω_3 are to each other, the smaller the "peak" in the phase curve becomes. For a proper PLL design it is important to know where the local maximum of the phase curve in the frequency range $\omega_2 < \omega < \omega_3$ really is. A simple computation shows that the phase function is maximum at a frequency ω_{opt}, which is the geometric mean of ω_2 and ω_3,[48]

$$\omega_{opt} = \sqrt{\omega_2 \omega_3} \tag{4.3}$$

We wanted to have sufficient phase margin at the transition frequency; it is therefore optimum to choose the transition frequency such that it is identical

with ω_{opt}. In our example we have chosen $\omega_2 = 31{,}600\,s^{-1}$ and $\omega_3 = 316{,}000\,s^{-1}$; hence ω_T is at $100{,}000\,s^{-1}$. In real higher-order PLL designs it has proved useful to set the ratio ω_3/ω_2 approximately to 10; when doing so, we get sufficient phase margin to provide stable operation.

The example in Fig. 4.1 was related to a third-order PLL. When building PLL with order higher than 3, additional poles (and eventually zeros) must be placed. In case of a fifth-order PLL, for example, the open-loop transfer function $G(s)$ has 5 poles and may also have up to 5 zeros. The gain roll-off at higher frequencies has a slope which is given by

$$S = 20(n_z - n_p) \qquad \text{dB} \tag{4.4}$$

where n_z = number of zeros and n_p = number of poles of $G(s)$. Because we want the gain to roll off with maximum slope, the number of zeros 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 of Fig. 4.1. If the system has higher order than 3, additional poles have to be placed at frequencies above ω_3 in Fig. 4.1. 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 ω_T. Consequently the loop design should start with an initial value for ω_T. Usually the designer of a PLL has an idea about the desired loop bandwidth. There are a number of parameters related to bandwidth, e.g., $\omega_{3\,dB}$ [the 3-dB closed-loop bandwidth of $H(\omega)$], B_L (the noise bandwidth of the loop), and ω_n (the natural frequency). All these parameters, including ω_T, are close together, i.e., have about the same order. Because the natural frequency is defined only for second-order systems (in a strict sense), it seems more practical to start by setting $\omega_{3\,dB}$. Often $\omega_{3\,dB}$ is chosen to be around 1/20 the (scaled-down) center frequency ω_0'. What we need now is a mathematical relation between $\omega_{3\,dB}$ and ω_T. For a second-order high-gain loop with $\zeta = 0.707$, it can be shown that $\omega_{3\,dB} \approx 2.06\,\omega_n$ and $\omega_T \approx 1.55\,\omega_n$. Hence we have

$$\omega_T \approx \omega_{3dB}/1.33 \tag{4.5}$$

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 so that its magnitude curve crosses the 0-dB line with a slope of about −20 dB/dec. The poles at higher frequencies than ω_T then have little influence on the Bode plot in the vicinity of ω_T; hence the ratio $\omega_{3\,dB}/\omega_T$ is mostly in the range 1.3 to 1.5.

In Sections 4.3 to 4.5 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. 5 a computer program will be presented that allows design of 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.

4.3 Designing Third-Order PLLs

In a third-order PLL, a second-order loop filter must be chosen. In this section we describe three variants of a second-order loop filter, a passive lead-lag, an active lead-lag, and an active PI filter. We start the discussion with the passive lead-lag.

4.3.1 Passive lead-lag loop filter

The schematic of this filter is shown in Fig. 4.2. Its transfer function is

$$F(s) = \frac{1 + s(\tau_2 + \tau_3)}{1 + s(\tau_1 + \tau_2 + \tau_3) + s^2\tau_1\tau_3} \tag{4.6}$$

with $\tau_1 = R_1C_1$, $\tau_2 = R_2C_1$, $\tau_3 = R_2C_2$. To place the poles and zeros of the PLL's open-loop transfer function $G(s)$, it is more convenient to write this function in factorized form:

$$F(s) = \frac{1 + sT_2}{(1 + sT_1)(1 + sT_3)} \tag{4.7}$$

The open-loop transfer function of the third-order PLL then becomes

$$G(s) = \frac{K_0K_d}{Ns} \frac{1 + sT_2}{(1 + sT_1)(1 + sT_3)} \tag{4.8}$$

To get the Bode diagram we set $s = j\omega$ and plot the magnitude $|G(\omega)|$ versus radian frequency ω. This is shown by Fig. 4.3. To determine the corner frequencies ω_1, ω_2, and ω_3, the transition frequency ω_T must be known first. It has proved 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 $\omega_{3\,\mathrm{dB}}$. A default value

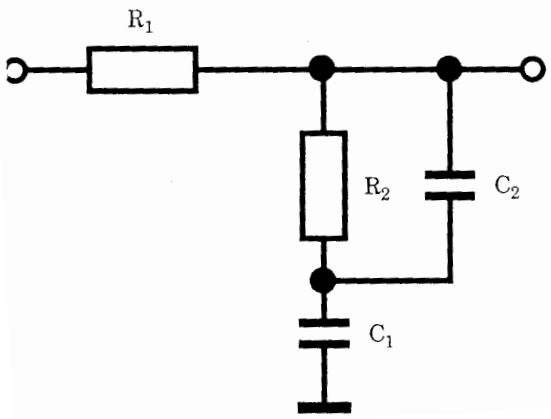


Figure 4.2 Second-order passive lead-lag loop filter.

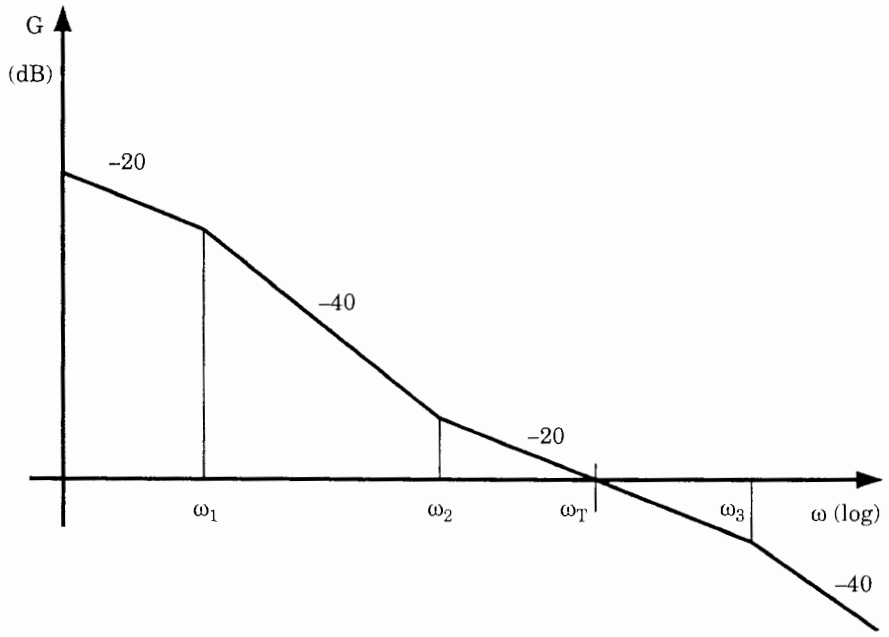


Figure 4.3 Bode diagram (magnitude plot) of a third-order PLL using the passive lead-lag loop filter.

could by $\omega_{3\,\mathrm{dB}} = 0.05 \cdot \omega_0'$, where ω_0' is the down-scaled center (radian) frequency of the PLL.

Step 2. Using the approximation in Eq. (4.5) we compute $\omega_T = \omega_{3\,\mathrm{dB}}/1.33$. With the passive lead-lag filter there is an upper limit for ω_T, however, which is given by

$$\omega_{T,\max} = \frac{K_0 K_d}{N} \tag{4.9}$$

When a larger value of ω_T is specified, this later results in a negative value for τ_1 [see Eq. (4.6)]; hence that filter would be unrealizable.

Step 3. Next the corner frequencies ω_2 and ω_3 are chosen. To get sufficient phase margin, it is recommended to set

$$\omega_3 = \omega_T \sqrt{10}$$
$$\omega_2 = \omega_T / \sqrt{10}$$

Step 4. Now the corner frequency ω_1 must be chosen such that the open-loop gain at transition frequency ω_T bcomes 1. This leads to

$$\omega_1 = \frac{K_0 K_d}{N \omega_T} \tag{4.10}$$

Step 5. The time constants T_1, T_2, T_3 are computed from $T_1 = 1/\omega_1$, $T_2 = 1/\omega_2$, $T_3 = 1/\omega_3$.

Step 6. Given T_1, T_2, T_3 we can calculate the filter time constants τ_1, τ_2, τ_3 now. By comparison of coefficients in Eqs. (4.6) and (4.7) we get

$$\begin{aligned} \tau_1 &= T_1 + T_3 - T_2 \\ \tau_2 &= T_2 - \frac{T_1 T_3}{T_1 + T_3 - T_2} \\ \tau_3 &= T_2 - \tau_2 \end{aligned} \tag{4.11}$$

Step 7. (Optional) There are cases where the designer modifies the values for τ_1, τ_2, τ_3 or adopts the values for τ_1, τ_2, τ_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, T_2, T_3, i.e., on the corner frequencies ω_1, ω_2, ω_3 in the magnitude plot. When solving the Eqs. (4.11) for T_1, T_2, T_3, we get

$$\begin{aligned} T_1 &= \frac{\tau_1 + \tau_2 + \tau_3 + \sqrt{(\tau_1 + \tau_2 + \tau_3)^2 - 4\tau_1\tau_3}}{2} \\ T_2 &= \tau_2 + \tau_3 \\ T_3 &= \frac{\tau_1 \tau_3}{T_1} \end{aligned} \tag{4.12}$$

Step 8. Finally the filter components (R, C) are computed from the values for τ_1, τ_2, τ_3, [see Eq. (4.6)]. The value of C_1 can be chosen arbitrarily. It should be set such that "reasonable" values for the resistors are obtained, i.e., in the region of about 1 to 100 kΩ.

4.3.2 Active lead-lag loop filter

The schematic of a second-order active lead-lag filter is shown in Fig. 4.4. The transfer function for this filter is given by

$$F(s) = K_a \frac{1 + s(\tau_2 + \tau_3)}{(1 + s\tau_1)(1 + s\tau_3)} \tag{4.13}$$

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

$$F(s) = K_a \frac{1 + sT_2}{(1 + sT_1)(1 + sT_3)} \tag{4.14}$$

The open-loop gain of the third-order PLL then reads

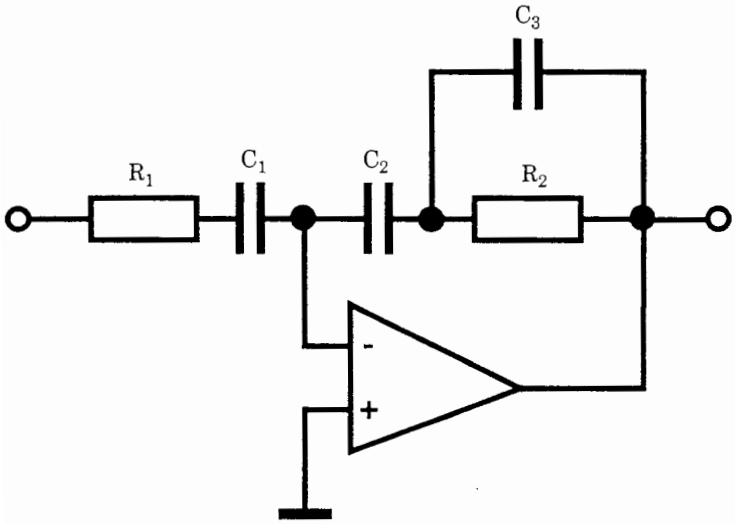


Figure 4.4 Schematic of the second-order active lead-lag filter.

$$G(s) = \frac{K_0 K_d K_a}{Ns} \frac{1 + sT_2}{(1 + sT_1)(1 + sT_3)} \tag{4.15}$$

For this design the procedure is very similar to that for the passive lead-lag filter. The Bode plot in Fig. 4.3 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 $\omega_{3\,\mathrm{dB}}$. A default value could be $\omega_{3\,\mathrm{dB}} = 0.05\omega_0'$, where ω_0' is the down-scaled center (radian) frequency of the PLL.

Step 2. Using the approximation in Eq. (4.5) we compute $\omega_T = \omega_{3\,\mathrm{dB}}/1.33$. In contrast to the passive lead-lag filter, there is no mathematical restriction on the size of ω_T here, because we have an additional parameter K_a, which always can be specified such that the open-loop gain becomes 1 at the transition frequency ω_T.

Step 3. Next the corner frequencies ω_2 and ω_3 are chosen. To get sufficient phase margin, it is recommended to set

$$\omega_3 = \omega_T \sqrt{10}$$
$$\omega_2 = \omega_T / \sqrt{10}$$

Step 4. It is recommended to set $\omega_1 = \omega_2/10$; thus the open-loop gain rolls off with −40 dB/dec over one full decade of frequency. K_a must be chosen such that the open-loop gain just becomes 1 at ω_T. For K_a we get

$$K_a = \frac{10N\omega_T}{K_0 K_d} \tag{4.16}$$

Step 5. The time constants T_1, T_2, T_3 are computed from $T_1 = 1/\omega_1$, $T_2 = 1/\omega_2$, $T_3 = 1/\omega_3$.

Step 6. Given T_1, T_2, T_3 we can calculate the filter time constants τ_1, τ_2, τ_3 now. By comparison of coefficients in Eqs. (4.13) and (4.14) we get

$$\begin{aligned} \tau_1 &= T_1 \\ \tau_2 &= T_2 - T_3 \\ \tau_3 &= T_3 \end{aligned} \tag{4.17}$$

Step 7. (Optional) There are cases where the designer modifies the values for τ_1, τ_2, τ_3 or adopts the values for τ_1, τ_2, τ_3 from an earlier design. When these parameters have been directly set or altered, it may be desirable to know also the values for T_1, T_2, T_3, i.e., the corner frequencies ω_1, ω_2, ω_3 in the magnitude plot. When solving the Eqs. (4.17) for T_1, T_2, T_3 we get

$$\begin{aligned} T_1 &= \tau_1 \\ T_2 &= \tau_2 + \tau_3 \\ T_3 &= \tau_3 \end{aligned} \tag{4.18}$$

Step 8. Finally the filter components (R, C) are computed from the values for τ_1, τ_2, τ_3. The value of C_1 can be chosen arbitrarily. It should be set such that "reasonable" values for the resistors are obtained, i.e., in the region of about 1 to 100 kΩ.

4.3.3 Active PI loop filter

The schematic of a second-order active PI filter is shown in Fig. 4.5. The transfer function of this filter is given by

$$F(s) = \frac{1 + s(\tau_2 + \tau_3)}{s\tau_1(1 + s\tau_3)} \tag{4.19}$$

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

$$F(s) = \frac{1 + sT_2}{sT_1(1 + sT_3)} \tag{4.20}$$

The open-loop gain of the third-order PLL then reads

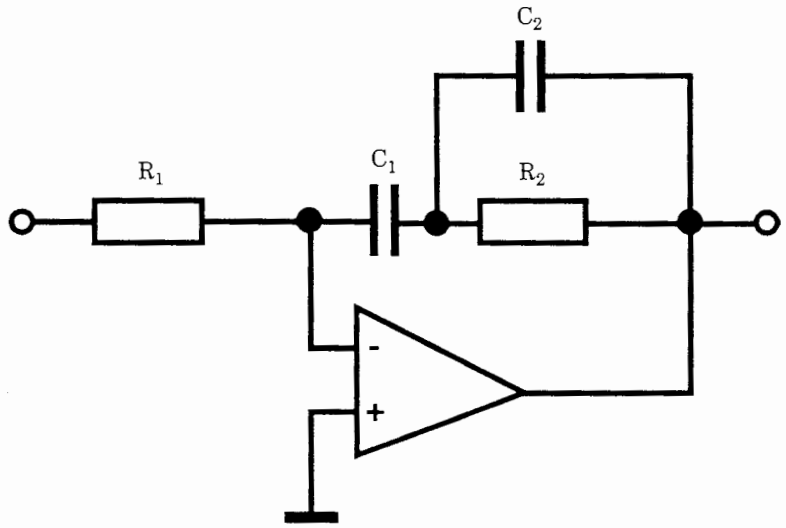


Figure 4.5 Schematic of second-order active PI loop filter.

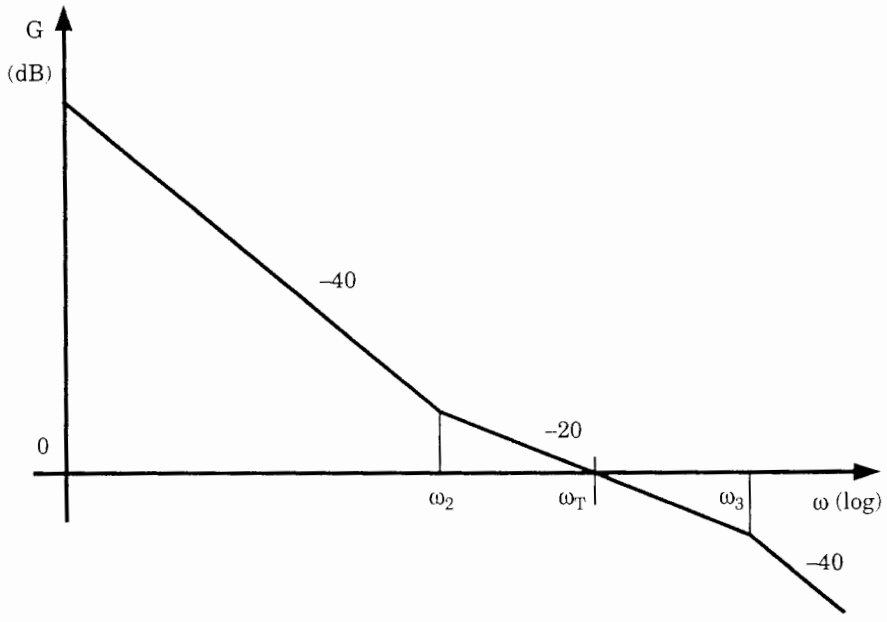


Figure 4.6 Bode plot of third-order PLL using the active PI loop filter.

$$G(s) = \frac{K_0 K_d}{Ns} \frac{1 + sT_2}{sT_1(1 + sT_3)} \tag{4.21}$$

The Bode plot (magnitude) for this system is shown in Fig. 4.6.

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 $\omega_{3\,\mathrm{dB}}$. A default value

could be $\omega_{3\,dB} = 0.05\omega_0'$, where ω_0' is the down-scaled center (radian) frequency of the PLL.

Step 2. Using the approximation in Eq. (4.5), we compute $\omega_T = \omega_{3\,dB}/1.33$. There is no mathematical restriction on the size of ω_T, because the parameter T_1 can be chosen such that the open-loop gain becomes 1 at $\omega = \omega_T$.

Step 3. Next the corner frequencies ω_2 and ω_3 are chosen. To get sufficient phase margin, it is recommended to set

$$\omega_3 = \omega_T\sqrt{10}$$
$$\omega_2 = \omega_T/\sqrt{10}$$

Step 4. Parameter T_1 is chosen such that the open-loop gain is 1 at $\omega = \omega_T$. This leads to

$$T_1 = \frac{K_0 K_d}{N\omega_2^2\sqrt{10}} \tag{4.22}$$

Step 5. The time constants T_2, T_3 are computed from $T_2 = 1/\omega_2$ and $T_3 = 1/\omega_3$.

Step 6. Given T_1, T_2, T_3 we can calculate the filter time constants τ_1, τ_2, t_3 now. By comparison of coefficients in Eqs. (4.19) and (4.20) we get

$$\begin{aligned}\tau_1 &= T_1\\ \tau_2 &= T_2 - T_3\\ \tau_3 &= T_3\end{aligned} \tag{4.23}$$

Step 7. (Optional) There are cases where the designer modifies the values for τ_1, τ_2, τ_3 or adopts the values for τ_1, τ_2, τ_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, T_2, T_3, i.e., on the corner frequencies ω_2 and ω_3 in the magnitude plot. When solving the Eqs. (4.23) for T_1, T_2, T_3 we get

$$\begin{aligned}T_1 &= \tau_1\\ T_2 &= \tau_2 + \tau_3\\ T_3 &= \tau_3\end{aligned} \tag{4.24}$$

Step 8. Finally the filter components (R, C) are computed from the values for τ_1, τ_2, τ_3. The value of C_1 can be chosen arbitrarily. It should be set such that "reasonable" values for the resistors are obtained, i.e., in the region of about 1 to 100 kΩ.

4.4 Designing Fourth-Order PLLs

To obtain a fourth-order PLL we must use a third-order loop filter. Of course such a filter can be realized as a passive lead-lag, an active lead-lag, or an active PI filter. All these filters will have three poles. It is common practice to build third-order filters that have a complex-conjugate pole pair. A passive lead-lag filter built from resistors and capacitors cannot realize complex poles, but all the poles are located on the negative real axis in the s plane. To get complex poles with this kind of filter, we would have to use inductors. Most designers try to avoid inductors in filter design and resort to active RC filters. For this reason we will drop the passive lead-lag filter and use active filters exclusively.

4.4.1 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. 4.7. 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 (see Chap. 2). Section 2 is a second-order low-pass 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

$$F(s) = K_a \frac{1+s\tau_2}{1+s\tau_1} \cdot \frac{1}{(1+s\tau_3)(1+s\tau_4)} \tag{4.25a}$$

with $\tau_1 = R_1C_1$, $\tau_2 = R_2C_2$, 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 $F(s)$:

$$F(s) = K_a \frac{1+s\tau_2}{1+s\tau_1} \cdot \frac{1}{1+\dfrac{2s\zeta_s}{\omega_s}+\dfrac{s^2}{\omega_s^2}} \tag{4.25b}$$

In the second case ω_s is the natural frequency and ζ_s the damping factor of the second-order frequency response. For the open-loop transfer function $G(s)$ of

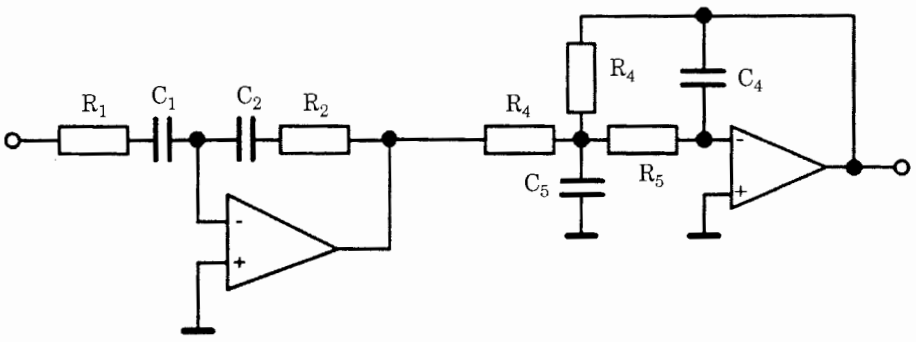


Figure 4.7 Third-order active lead-lag filter.

the fourth-order PLL we get two versions, one for real poles and one for complex poles. For real poles, $G(s)$ reads

$$G(s) = \frac{K_0 K_d K_a}{Ns} \cdot \frac{1+s\tau_2}{1+s\tau_1} \cdot \frac{1}{(1+s\tau_3)(1+s\tau_4)} \tag{4.26a}$$

For complex poles, $G(s)$ reads

$$G(s) = \frac{K_0 K_d K_a}{Ns} \cdot \frac{1+s\tau_2}{1+s\tau_1} \cdot \frac{1}{1+\dfrac{2s\zeta_s}{\omega_s}+\dfrac{s^2}{\omega_s^2}} \tag{4.26b}$$

Let us look at the magnitude plot now, Fig. 4.8. 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 ω_1 and ω_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 ω_3 and ω_4, as Fig. 4.8 shows. At ω_3 and at ω_4 the slope of the magnitude curve increases by –20 dB/dec. When the poles of the second filter section are complex, however, there is only one additional break point at ω_s. At this point the slope increases by –40 dB/dec.

We now have to provide a procedure for placing the poles and the zero of the third-order filter. It includes the following steps:

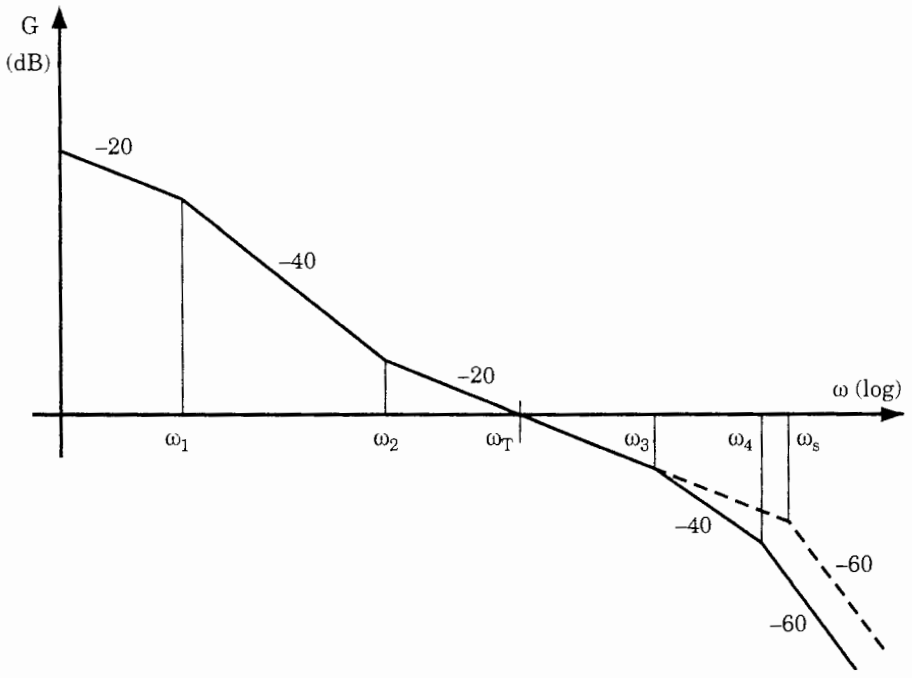


Figure 4.8 Bode plot for the fourth-order PLL. The solid curve represents the case where the poles of the second-order filter are real. The dashed curve shows the case where these poles form a complex pair.

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 $\omega_{3\,dB}$. A default value could be $\omega_{3\,dB} = 0.05\omega_0'$, where ω_0' is the down-scaled center (radian) frequency of the PLL.

Step 2. Using the approximation in Eq. (4.5) we compute $\omega_T = \omega_{3\,dB}/1.33$. There is no mathematical restriction on the size of ω_T here, because we have an additional parameter K_a, which always can be specified such that the open-loop gain becomes 1 at the transition frequency ω_T.

Step 3. Next the corner frequency ω_2 is chosen. To get sufficient phase margin, it is recommended to set

$$\omega_2 = \omega_T/\sqrt{10}$$

Step 4. It is recommended to set $\omega_1 = \omega_2/10$; thus the open-loop gain rolls off with −40 dB/dec over one full decade of frequency. K_a must be chosen such that the open-loop gain just becomes 1 at ω_T. For K_a we get

$$K_a = \frac{10N\omega_T}{K_0K_d} \tag{4.27}$$

Step 5. The time constants τ_1 and τ_2 are computed from $\tau_1 = 1/\omega_1$, $\tau_2 = 1/\omega_2$.

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 6, otherwise with step 8.

Step 6. (Complex poles) We must choose a suitable value for natural frequency ω_s and damping factor ζ_s. For ζ_s it is optimum to set $\zeta_s = 0.707$. Experience shows that sufficient phase margin is obtained if ω_s is placed at $5\omega_T$; hence we set $\omega_s = 5\omega_T$.

Step 7. Given ω_s and ζ_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 arbitarily. It should be set such that we obtain "reasonable" values for the resistors, i.e., in the range 1 to 100 kΩ.

The remaining components are computed from

$$\begin{aligned} C_5 &= \frac{2C_4}{\zeta_s^2} \\ R_4 &= \sqrt{\frac{2}{\omega_s^2 C_4 C_5}} \\ R_5 &= \frac{R_4}{2} \end{aligned} \tag{4.28}$$

Step 8. (Real poles) We must choose values for the break frequencies ω_3 and ω_4 now; see Fig. 4.8. It is recommended to set $\omega_3 = \omega_T\sqrt{10}$ 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. (4.25*a*) into the filter function given in Eq. (4.25*b*); i.e., we would compute ω_s and ζ_s from given ω_3 and ω_4. Comparing coefficients in Eqs. (4.25*a*) and (4.25*b*) yields

$$\omega_s = \frac{1}{\sqrt{\tau_3\tau_4}}$$
$$\zeta_s = \frac{\tau_3 + \tau_4}{2\sqrt{\tau_3\tau_4}} \tag{4.29}$$

Given ω_s and ζ_s, we now can compute the R and C values using Eqs. (4.28).

4.4.2 Active PI loop filter

Third-order active PI filters can be implemented in many ways. One version is shown in Fig. 4.9. 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

$$F(s) = \frac{1 + s\tau_2}{s\tau_1} \cdot \frac{1}{(1 + s\tau_3)(1 + s\tau_4)} \tag{4.30a}$$

with $\tau_1 = R_1C_1$ and $\tau_2 = R_2C_1$. When the poles of the second filter section form a complex pair, it is more practical to use a different notation for $F(s)$:

$$F(s) = \frac{1 + s\tau_2}{s\tau_1} \cdot \frac{1}{1 + \dfrac{2s\zeta_s}{\omega_s} + \dfrac{s^2}{\omega_s^2}} \tag{4.30b}$$

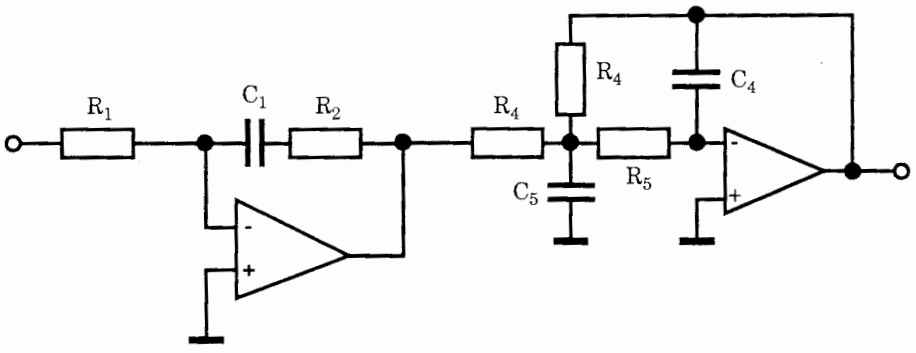


Figure 4.9 Third-order active PI loop filter.

In the second case ω_s is the natural frequency and ζ_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 real poles and one for complex poles. For real poles, $G(s)$ reads

$$G(s) = \frac{K_0 K_d}{Ns} \cdot \frac{1 + s\tau_2}{s\tau_1} \cdot \frac{1}{(1 + s\tau_3)(1 + s\tau_4)} \tag{4.31a}$$

For complex poles, $G(s)$ reads

$$G(s) = \frac{K_0 K_d}{Ns} \cdot \frac{1 + s\tau_2}{s\tau_1} \cdot \frac{1}{1 + \dfrac{2s\zeta_s}{\omega_s} + \dfrac{s^2}{\omega_s^2}} \tag{4.31b}$$

Let us have a look on the magnitude plot now, Fig. 4.10. 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 ω_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 ω_3 and ω_4, as in Fig. 4.10. At ω_3 and at ω_4 the slope of the magnitude curve increases by –20 dB/dec. When the poles of

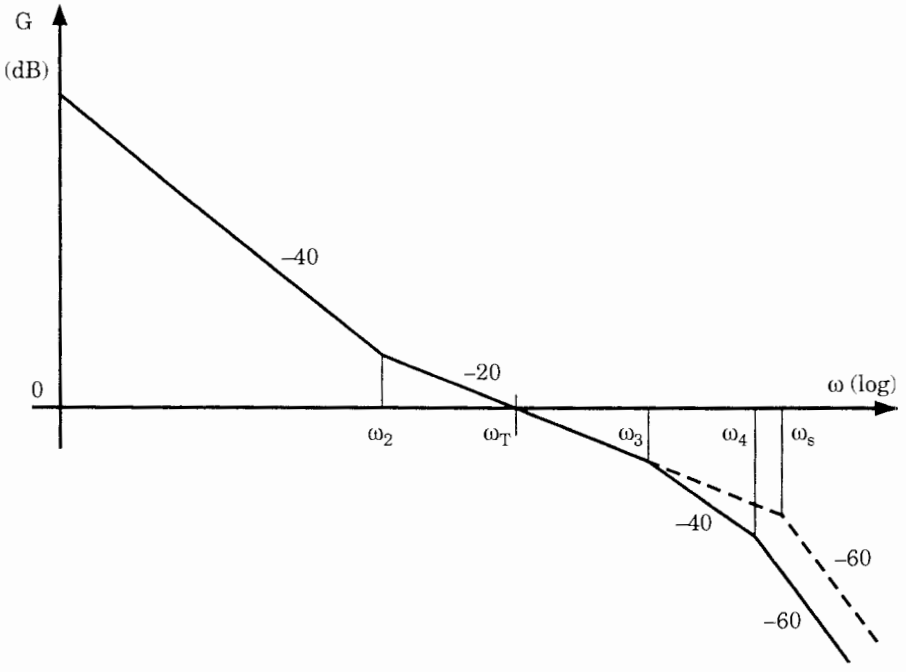


Figure 4.10 Bode plot (magnitude) of fourth-order PLL. The solid curve represents the case where the poles of the second-order filter are real. The dashed curve shows the case where these poles form a complex pair.

the second filter section are complex, however, there is only one additional break point at ω_s. At this point the slope increases by –40 dB/dec.

We now have to 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 $\omega_{3\,\mathrm{dB}}$. A default value could be $\omega_{3\,\mathrm{dB}} = 0.05\omega_0'$, where ω_0' is the down-scaled center (radian) frequency of the PLL.

Step 2. Using the approximation in Eq. (4.5), we compute $\omega_T = \omega_{3\,\mathrm{dB}}/1.33$. There is no mathematical restriction on the size of ω_T here, because time constant τ_1 can be set such that the open-loop gain becomes 1 at the transition frequency ω_T.

Step 3. Next the corner frequency ω_2 is chosen. To get sufficient phase margin, it is recommended to set

$$\omega_2 = \omega_T/\sqrt{10}$$

Step 4. Parameter τ_1 is chosen such that the open-loop gain is 1 at $\omega = \omega_T$. This leads to

$$\tau_1 = \frac{K_0 K_d}{N\omega_2^2\sqrt{10}} \tag{4.32}$$

Step 5. Time constant τ_2 is computed from $\tau_2 = 1/\omega_2$.

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 6, otherwise with step 8.

Step 6. (Complex poles) We must choose a suitable value for natural frequency ω_s and damping factor ζ_s. For ζ_s it is optimum to set $\zeta_s = 0.707$. Experience shows that sufficient phase margin is obtained if ω_s is placed at $5\omega_T$; hence we set $\omega_s = 5\omega_T$.

Step 7. Given ω_s and ζ_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 arbitarily. It should be set such that we obtain "reasonable" values for the resistors, i.e., in the range 1 to 100 kΩ.

The remaining components are computed from

$$C_5 = \frac{2C_4}{\zeta_s^2}$$
$$R_4 = \sqrt{\frac{2}{\omega_s^2 C_4 C_5}} \tag{4.33}$$
$$R_5 = \frac{R_4}{2}$$

Step 8. (Real poles) We must choose values for the break frequencies ω_3 and ω_4 now; see Fig. 4.10. It is recommended to set $\omega_3 = \omega_T\sqrt{10}$ 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. (4.30*a*) into to filter function given in Eq. (4.30*b*); i.e., we would compute ω_s and ζ_s from given ω_3 and ω_4. Comparing coefficients in Eqs. (4.30*a*) and (4.30*b*) yields

$$\omega_s = \frac{1}{\sqrt{\tau_3\tau_4}}$$
$$\zeta_s = \frac{\tau_3 + \tau_4}{2\sqrt{\tau_3\tau_4}} \tag{4.34}$$

Given ω_s and ζ_s, we now can compute the R and C values using Eqs. (4.33).

4.5 Designing Fifth-Order PLLs

When designing a fifth-order PLL, we must select a fourth-order loop filter. As in case of the fourth-order PLL, we use active filters exclusively. The procedure is simular to that applied for the fourth-order PLL.

4.5.1 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. 4.11. 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 $s = -\omega_3$; the zero is at $s = -\omega_2$. Section 2 is a second-order low-pass 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

$$F(s) = K_a \frac{1 + s(\tau_2 + \tau_3)}{(1 + s\tau_1)(1 + s\tau_3)} \cdot \frac{1}{(1 + s\tau_4)(1 + s\tau_5)} \tag{4.35a}$$

with $\tau_1 = R_1C_1$, $\tau_2 = R_2C_2$, $\tau_3 = R_2C_3$, 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

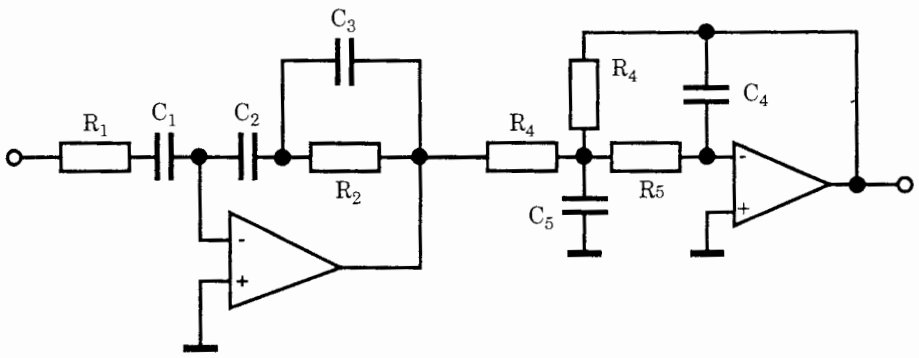


Figure 4.11 Fourth-order active lead-lag filter.

$$F(s) = K_a \frac{1 + s(\tau_2 + \tau_3)}{(1 + s\tau_1)(1 + s\tau_3)} \cdot \frac{1}{1 + \dfrac{2s\zeta_s}{\omega_s} + \dfrac{s^2}{\omega_s^2}} \tag{4.35b}$$

In the second case ω_s is the natural frequency and ζ_s 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. (4.35*a*) and (4.35*b*) in standardized form. For the real poles we get

$$F(s) = K_a \frac{1 + sT_2}{(1 + sT_1)(1 + sT_3)} \cdot \frac{1}{(1 + s\tau_4)(1 + s\tau_5)} \tag{4.36a}$$

and for the complex poles we have

$$F(s) = K_a \frac{1 + sT_2}{(1 + sT_1)(1 + sT_3)} \cdot \frac{1}{1 + \dfrac{2s\zeta_s}{\omega_s} + \dfrac{s^2}{\omega_s^2}} \tag{4.36b}$$

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

$$G(s) = \frac{K_0 K_d K_a}{Ns} \cdot \frac{1 + sT_2}{(1 + sT_1)(1 + sT_3)} \cdot \frac{1}{(1 + s\tau_4)(1 + s\tau_{54})} \tag{4.36c}$$

For complex poles, $G(s)$ reads

$$G(s) = \frac{K_0 K_d K_a}{Ns} \cdot \frac{1 + sT_2}{(1 + sT_1)(1 + sT_3)} \cdot \frac{1}{1 + \dfrac{2s\zeta_s}{\omega_s} + \dfrac{s^2}{\omega_s^2}} \tag{4.36d}$$

Let us look at the magnitude plot now, Fig. 4.12. The first filter section has poles at $s = -\omega_1$ and $s = -\omega_3$ and a zero at $s = -\omega_2$. It therefore provides the

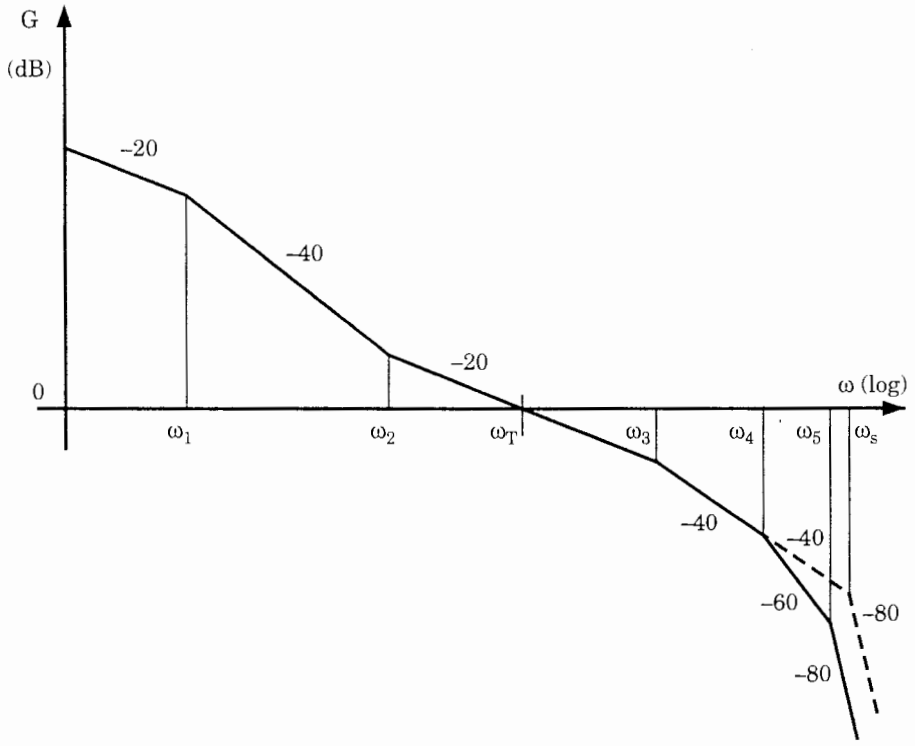


Figure 4.12 Bode plot for the fifth-order PLL. The solid curve represents the case where the poles of the second filter section are real. The dashed curve shows the case where these poles form a complex pair.

breakpoints of the asymptotic magnitude plot at ω_1, ω_2, and ω_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 ω_4 and ω_5, as in Fig. 4.12. At ω_4 and at ω_5 the slope of the magnitude curve increases by −20 dB/dec. When the poles of the second filter section are complex, however, there is only one additional breakpoint, at ω_s. At this point the slope increases by −40 dB/dec.

We now have to 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 should be the bandwidth of the PLL; hence it is reasonable to start by choosing a value for $\omega_{3\,\mathrm{dB}}$. A default value could be $\omega_{3\,\mathrm{dB}} = 0.05\omega_0'$, where ω_0' is the down-scaled center (radian) frequency of the PLL.

Step 2. Using the approximation in Eq. (4.5) we compute $\omega_T = \omega_{3\,\mathrm{dB}}/1.33$. There is no mathematical restriction on the size of ω_T, because we have an additional parameter K_a, which always can be specified such that the open-loop gain becomes 1 at the transition frequency ω_T.

Step 3. Next the corner frequencies ω_2 and ω_3 are chosen. To get sufficient phase margin, it is recommended to set

$$\omega_2 = \omega_T/\sqrt{10}$$
$$\omega_3 = \omega_T\sqrt{10}$$

Step 4. It is recommended to set $\omega_1 = \omega_2/10$; thus the open-loop gain rolls off with −40 dB/dec over one full decade of frequency. K_a must be chosen such that the open-loop gain just becomes 1 at ω_T. For K_a we get

$$K_a = \frac{10N\omega_T}{K_0K_d} \tag{4.37}$$

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 τ_1, τ_2, and τ_3. They are computed from

$$\begin{aligned}\tau_1 &= T_1\\ \tau_2 &= T_2 - T_3\\ \tau_3 &= T_3\end{aligned} \tag{4.38}$$

Step 7. (Optional) Sometimes the designer alters the values τ_1, τ_2, τ_3 and wants to know what is the effect on the quantities T_1, T_2, T_3. The variables T_1, T_2, T_3 are then computed from τ_1, τ_2, τ_3 as follows:

$$\begin{aligned}T_1 &= \tau_1\\ T_2 &= \tau_2 + \tau_3\\ T_3 &= \tau_3\end{aligned} \tag{4.39}$$

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 ω_s and damping factor ζ_s. For ζ_s it is optimum to set $\zeta_s = 0.707$. Experience shows that sufficient phase margin is obtained if ω_s is placed at $5\omega_T$; hence we set $\omega_s = 5\omega_T$.

Step 9. Given ω_s and ζ_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 arbitarily. It should be set such that we obtain "reasonable" values for the resistors, i.e., in the range 1 to 100 kΩ.

The remaining components are computed from

$$C_5 = \frac{2C_4}{\zeta_s^2}$$
$$R_4 = \sqrt{\frac{2}{\omega_s^2 C_4 C_5}} \tag{4.40}$$
$$R_5 = \frac{R_4}{2}$$

Step 10. (Real poles) We must choose values for the break frequencies ω_4 and ω_5 now; see Fig. 4.12. It is recommended to set $\omega_4 = 5\omega_3$ and $\omega_5 = 5\omega_4$. 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. (4.35*a*) into the filter function given in Eq. (4.35*b*); i.e., we would compute ω_s and ζ_s from given ω_4 and ω_5. Comparing coefficients in Eqs. (4.35*a*) and (4.35*b*) yields

$$\omega_s = \frac{1}{\sqrt{\tau_4 \tau_5}}$$
$$\zeta_s = \frac{\tau_4 + \tau_5}{2\sqrt{\tau_4 \tau_5}} \tag{4.41}$$

Given ω_s and ζ_s, we now can compute the R and C values using Eqs. (4.40).

4.5.2 Active PI loop filter

Fourth-order active PI filters can be implemented in many ways. One version is shown in Fig. 4.13. The filter consists of two sections. Section 1 is a second-order PI filter. It has two poles, one 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

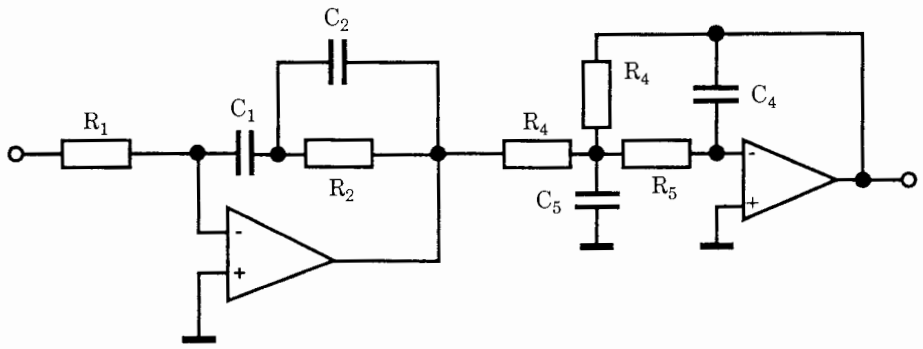


Figure 4.13 Fourth-order active PI loop filter.

second-order low-pass 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

$$F(s) = \frac{1 + s(\tau_2 + \tau_3)}{s\tau_1(1 + s\tau_3)} \cdot \frac{1}{(1 + s\tau_4)(1 + s\tau_5)} \tag{4.42a}$$

with $\tau_1 = R_1C_1$, $\tau_2 = R_2C_2$, and $\tau_3 = R_2C_3$. When the poles of the second filter section form a complex pair, it is more practical to use a different notation for $F(s)$

$$F(s) = \frac{1 + s(\tau_2 + \tau_3)}{s\tau_1(1 + s\tau_3)} \cdot \frac{1}{1 + \dfrac{2s\zeta_s}{\omega_s} + \dfrac{s^2}{\omega_s^2}} \tag{4.42b}$$

In the second case ω_s is the natural frequency and ζ_s 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. (4.42*a*) and (4.42*b*) in standardized form. For the real poles we get

$$F(s) = \frac{1 + sT_2}{sT_1(1 + sT_3)} \cdot \frac{1}{(1 + s\tau_4)(1 + s\tau_5)} \tag{4.43a}$$

and for the complex poles we have

$$F(s) = \frac{1 + sT_2}{sT_1(1 + sT_3)} \cdot \frac{1}{1 + \dfrac{2s\zeta_s}{\omega_s} + \dfrac{s^2}{\omega_s^2}} \tag{4.43b}$$

For the open-loop transfer function $G(s)$ of the fourth-order PLL, we get two versions, one for the real poles and one for the complex poles. For real poles, $G(s)$ reads

$$G(s) = \frac{K_0K_d}{Ns} \cdot \frac{1 + sT_2}{sT_1(1 + sT_3)} \cdot \frac{1}{(1 + s\tau_4)(1 + s\tau_{54})} \tag{4.44a}$$

For complex poles, $G(s)$ reads

$$G(s) = \frac{K_0K_d}{Ns} \cdot \frac{1 + sT_2}{sT_1(1 + sT_3)} \cdot \frac{1}{1 + \dfrac{2s\zeta_s}{\omega_s} + \dfrac{s^2}{\omega_s^2}} \tag{4.44b}$$

Let us look at the magnitude plot now, Fig. 4.14. 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 ω_2 and ω_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 ω_4 and ω_5, as shown in Fig. 4.14. At ω_4 and at ω_5 the slope of the magnitude curve increases by –20 dB/dec.

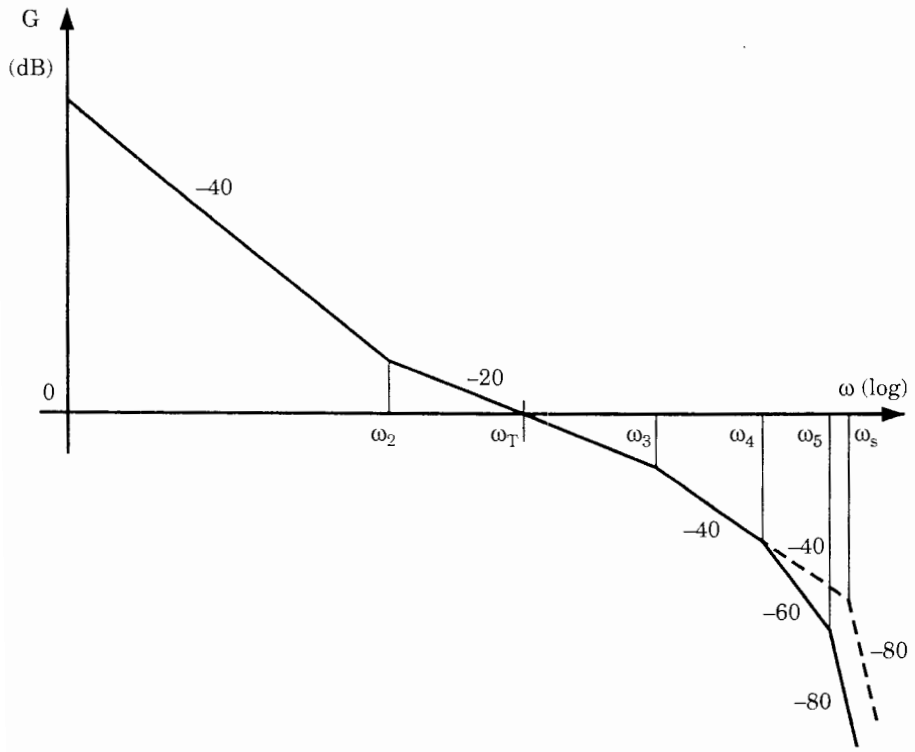


Figure 4.14 Bode plot (magnitude) of fifth-order PLL. The solid curve represents the case where the poles of the second filter section are real. The dashed curve shows the case where these poles form a complex pair.

When the poles of the second filter section are complex, however, there is only one additional breakpoint at ω_s. At this point the slope increases by –40 dB/dec.

We now have to 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 $\omega_{3\,\mathrm{dB}}$. A default value could be $\omega_{3\,\mathrm{dB}} = 0.05\omega_0'$, where ω_0' is the down-scaled center (radian) frequency of the PLL.

Step 2. Using the approximation in Eq. (4.5) we compute $\omega_T = \omega_{3\,\mathrm{dB}}/1.33$. There is no mathematical restriction on the size of ω_T here, because time constant T_1 can be set such that the open-loop gain becomes 1 at the transition frequency ω_T.

Step 3. Next, the corner frequencies ω_2 are chosen. To get sufficient phase margin, it is recommended to set

$$\omega_2 = \omega_T/\sqrt{10}$$
$$\omega_3 = \omega_T\sqrt{10}$$

Step 4. Parameter T_1 is chosen such that the open-loop gain is 1 at $\omega = \omega_T$. This leads to

$$T_1 = \frac{K_0 K_d}{N\omega_2^2\sqrt{10}} \tag{4.45}$$

Step 5. Time constants T_2 and T_3 are computed from $T_2 = 1/\omega_2$, $T_3 = 1/\omega_3$.

Step 6. To realize section 1 of the loop filter, we must determine the values τ_1, τ_2, τ_3. These are computed from T_1, T_2, T_3 by

$$\begin{aligned} \tau_1 &= T_1 \\ \tau_2 &= T_2 - T_3 \\ \tau_3 &= T_3 \end{aligned} \tag{4.46}$$

Step 7. (Optional) Sometimes the designer changes the values τ_1, τ_2, τ_3 and wants to know how the values T_1, T_2, T_3 are altered. Then T_1, T_2, T_3 can be computed from τ_1, τ_2, τ_3 from

$$\begin{aligned} T_1 &= \tau_1 \\ T_2 &= \tau_2 + \tau_3 \\ T_3 &= \tau_3 \end{aligned} \tag{4.47}$$

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 ω_s and damping factor ζ_s. For ζ_s it is optimum to set $\zeta_s = 0.707$. Experience shows that sufficient phase margin is obtained if ω_s is placed at $5\omega_T$; hence we set $\omega_s = 5\omega_T$.

Step 9. Given ω_s and ζ_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 arbitarily. It should be set such that we obtain "reasonable" values for the resistors, i.e., in the range 1 to 100 kΩ.

The remaining components are computed from

$$\begin{aligned} C_5 &= \frac{2C_4}{\zeta_s^2} \\ R_4 &= \sqrt{\frac{2}{\omega_s^2 C_4 C_5}} \\ R_5 &= \frac{R_4}{2} \end{aligned} \tag{4.48}$$

Step 10. (Real poles) We must choose values for the break frequencies ω_3 and ω_4 now; see Fig. 4.14. It is recommended to set $\omega_4 = 5\omega_3$ and $\omega_5 = 5\omega_4$. 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. (4.43*a*) into the filter function given in Eq. (4.43*b*); i.e., we would compute ω_s and ζ_s from given ω_4 and ω_5. Comparing coefficients in Eqs. (4.43*a*) and (4.43*b*) yields

$$\omega_s = \frac{1}{\sqrt{\tau_4 \tau_5}}$$
$$\zeta_s = \frac{\tau_4 + \tau_5}{2\sqrt{\tau_4 \tau_5}} \tag{4.49}$$

Given ω_s and ζ_s, we now can compute the R and C values using Eqs. (4.48).

4.6 The Key Parameters of Higher-Order PLLs

In Chap. 2 we derived a number of equations for computing parameters such as lock-in range and pull-out range for second-order PLLs. The question arises now how these parameters change with higher-order loops. As we have seen the key parameters depend on quantities such as natural frequency, damping factor, and gain factors such as K_d and K_0.

When checking the magnitude plots for third- and higher-order PLLs we note that there are poles and zeros located at frequencies where the open-loop gain is greater than 1; as an example that applies to the corner frequencies ω_1 and ω_2 in Fig. 4.3 (third-order loop). The additional poles are placed at frequencies where the open-loop gain is markedly less than 1 (as in the corner at ω_3 in Fig. 4.3). These higher-frequency poles have therefore little impact onto the transient behavior of the loop, hence 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 2.1 to 2.4 to compute key parameters such as lock-in range and lock-in time.

Although terms such as *natural frequency* ω_n and *damping factor* ζ 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.

4.7 Loop Filters for Phase Detectors with Charge Pump Output

4.7.1 Loop filters for second-order PLLs

In Sec. 2.7 we discussed phase-frequency detectors having a charge pump (current) output. We recognized that the loop filter cascaded with a charge pump PFD looks somewhat different from a loop filter that is used for voltage output PFDs; check the circuit in Fig. 2.40*c*, for example. In this section we will consider examples of higher-order loop filters suitable for cascading with charge pump PFDs. To start, let us review briefly the configuration used in Fig. 2.40*c* by the model in Fig. 4.15. This represents a passive first-order loop filter used in a second-order PLL.

In this block diagram the charge pump PFD has been split into two blocks:

- A conventional (voltage output) PFD
- A voltage-to-current converter

The output signal of the PFD is given by the current i_{out}, of course. The signal u_d is fictive only and will be used for the following computations. The voltage signal u_d would be given by

$$u_d = K_d \theta_e \tag{4.50}$$

where the phase detector gain K_d is computed as usual from Eq. (2.24) or (2.25). For the output current i_{out} we have

$$i_{\text{out}} = u_d / R_b$$

where R_b is the transimpedance of the voltage-to-current converter, as explained in Sec. 2.7. As noted there, R_b is usually set by an external resistor. We now express the loop filter output signal u_f as a function of the (fictive) u_d signal, rather than as a function of the current signal i_{out}, and obtain

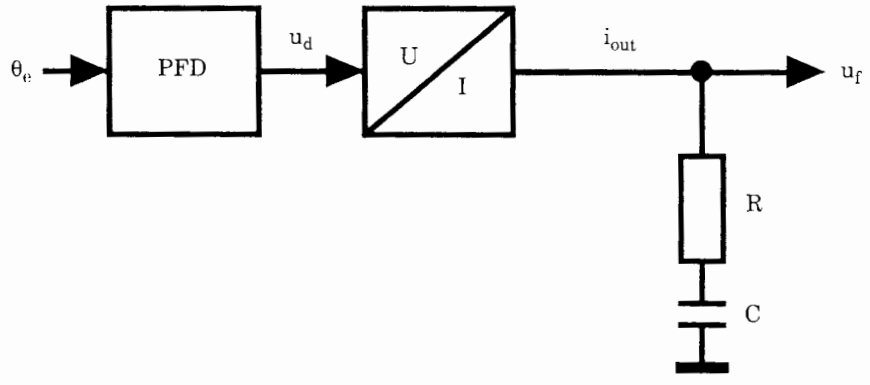


Figure 4.15 Model for a charge pump output PFD cascaded with a passive first-order lead-lag filter.

$$F(s) = \frac{U_f(s)}{U_d(s)} = \frac{1 + sRC}{sR_bC} \tag{4.51}$$

If u_f were computed as a function of i_{out}, however, the filter transfer function would have the dimension ohm, which is not very practical for numerical computations. By computing u_f from the fictive signal u_d, the transfer function $F(s)$ becomes dimensionless. Setting $\tau_1 = R_bC$ and $\tau_2 = RC$, $F(s)$ becomes

$$F(s) = \frac{1 + s\tau_2}{s\tau_1}$$

This is identical with the transfer function of the active PI loop filter [Eq. (2.28)]. In the following we will discuss loop filters of order 2 to 4, which can be used in combination with charge pump outputs. The model shown in Fig. 4.15 will be used for all computations.

4.7.2 Loop filters for third-order PLLs

A passive loop filter of order 2 is shown in Fig. 4.16. In a computation similar to that for the first-order loop filter, the filter transfer function $F(s)$ becomes

$$F(s) = \frac{1 + s\tau_2}{s\tau_1(1 + s\tau_3)}$$

with

$$\begin{aligned} \tau_1 &= R_b(C_1 + C_2) \\ \tau_2 &= RC_1 \\ \tau_3 &= R\frac{C_1C_2}{C_1 + C_2} \end{aligned} \tag{4.52}$$

As we have seen in Sec. 4.3, the time constant τ_3 is usally much smaller than τ_2, typically by a factor of 10; hence C_2 is much smaller than C_1. Under this condition, the expressions in Eq. (4.52) can be simplified to read

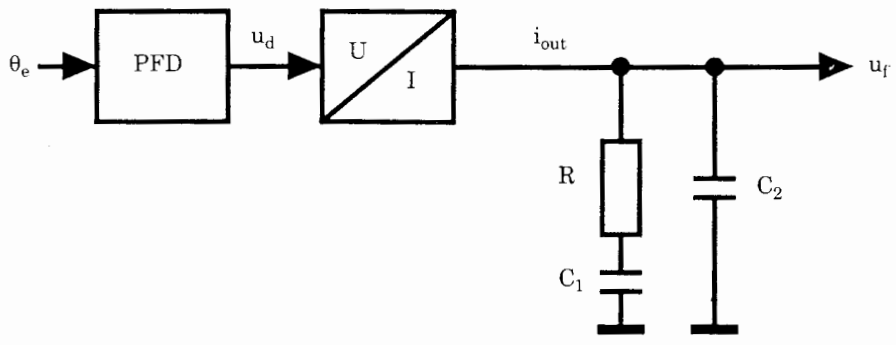


Figure 4.16 Passive second-order loop filter to be used with charge pump PFD.

$$\tau_1 \approx R_b C_1$$
$$\tau_2 \approx RC_1$$
$$\tau_3 \approx RC_2$$

The loop filter transfer function $F(s)$ is identical with that of the third-order PLL using the active PI loop filter, Eq. (4.20). The loop filter considered here can therefore be designed exactly like the second-order active PI filter.

4.7.3 Loop filters for fourth-order PLLs

A third-order passive loop filter is shown in Fig. 4.17. Its transfer function $F(s)$ is approximately given by

$$F(s) = \frac{1 + s\tau_2}{s\tau_1(1 + s\tau_3)(1 + s\tau_4)}$$

with

$$\begin{aligned} &\tau_2 > \tau_3 > \tau_4 \\ &\tau_1 \approx R_b C_1 \\ &\tau_2 \approx R_1 C_1 \\ &\tau_3 \approx R_1 C_3 \\ &\tau_4 \approx R_2 C_2 \end{aligned} \qquad (4.53a)$$

The approximation is valid under the assumptions

$$\tau_2 \gg \tau_3$$
$$\tau_3 \gg \tau_4$$

which are fulfilled in most applications. With this kind of filter, only real poles of $F(s)$ can be created. If complex poles are desired, the first-order passive loop filter shown in Fig. 4.15 could be cascaded with an active second-order filter as shown in the second section of the third-order filter in Fig. 4.9, for example.

Because the filter transfer function is identical with that of the fourth-order PLL using a third-order active PI filter [Eq. (4.30a)], this loop filter can be designed by using the design procedure given in Sec. 4.4.

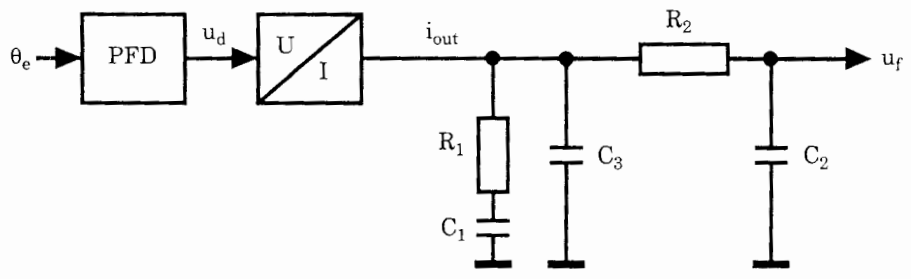


Figure 4.17 Third-order passive loop filter to be used with charge pump PFD.

4.7.4 Loop filters for fifth-order PLLs

A third-order passive loop filter is shown in Fig. 4.18. Its transfer function $F(s)$ is approximately given by

$$F(s) = \frac{1 + s\tau_2}{s\tau_1(1 + s\tau_3)(1 + s\tau_4)(1 + s\tau_5)}$$

with

$$\begin{aligned} &\tau_2 > \tau_3 > \tau_4 > \tau_5 \\ &\tau_1 \approx R_b C_1 \\ &\tau_2 \approx R_1 C_1 \\ &\tau_3 \approx R_1 C_3 \\ &\tau_4 \approx R_2 C_2 \\ &\tau_5 \approx R_4 C_4 \end{aligned} \tag{4.53b}$$

The approximation is valid under the assumptions

$$\begin{aligned} &\tau_2 \gg \tau_3 \\ &\tau_3 \gg \tau_4 \\ &\tau_4 \gg \tau_5 \end{aligned}$$

which are fulfilled in most applications. With this kind of filter, only real poles of $F(s)$ can be created. If complex poles are desired, the second-order passive loop filter shown in Fig. 4.16 could be cascaded with an active second-order filter, as shown in the second section of the third-order filter in Fig. 4.9, for example.

Because the filter transfer function is identical with that of the fifth-order PLL using a fourth-order active PI filter [Eq. (4.42*a*)], this loop filter can be designed using the design procedure given in Sec. 4.5.

A large number of passive and active loop filters that can be cascaded with charge pump PFDs are listed in the EasyPLL design program, a Web application created by National Semiconductor.[50]

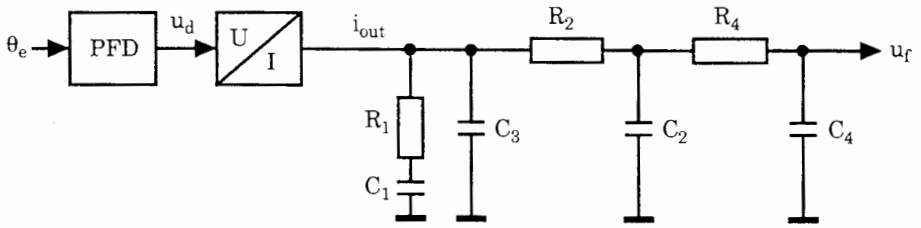


Figure 4.18 Fourth-order passive loop filter to be used with charge pump PFD.

Chapter

5

Computer-Aided Design and Simulation of Mixed-Signal PLLs

5.1 Overview

A CD is distributed with this book containing a program developed by the author that enables the user to design and simulate PLLs of different kinds. 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, start Windows Explorer, click on the D:\ drive (it is assumed that the CD uses drive D) and double click on the SETUP.EXE file in the right pane of Windows Explorer. During installation follow the instructions on screen.

The program can be started now. On start-up it displays a PLL block diagram which defines the symbols used throughout this application. In the toolbar (on top of the screen) there are a number of speedbuttons that look like this:

These buttons are used to call a number of options. Clicking the *CONFIG* button opens a configuration dialog (Fig. 5.1). This dialog box will be used to specify which category of PLL will be designed (LPLL, DPLL, or ADPLL). Moreover the user will have to tell the program which types of phase detectors, loop filters, etc., will be used in a particular configuration. Hitting the *DESIGN* speed button starts another dialog, the filter design dialog box (Fig. 5.2). Here the user will have to enter key parameters such as 3-dB bandwidth (f_{3dB}) and the like. This dialog box is used to compute the parameters of the loop filter (e.g., time constants τ_1, τ_2). With the *SIM* (simulation) speedbutton the user enters simulation mode (Fig. 5.5). Here the response of the loop to phase and frequency steps applied to the reference input can be simulated. The dialog box also allows you to add noise to the reference signal. By the *OPT* (options) speed-

button, you start an options dialog where you 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, etc., and fonts, font size, font style (italic, bold), etc. The speedbutton with the interrogation mark starts a *HELP* file, which provides plenty of information on this program and in addition an overview on PLL fundamentals and much more. The last speedbutton (*EXIT*) closes the program.

To see what the program can do, it is recommended that you start the program and perform a quick tour.

5.2 Quick Tour

Start the program. This will bring up the block diagram of the PLL. On top of the screen there is a toolbar with six speedbuttons labeled *CONFIG*, *DESIGN*, etc. Hit the *CONFIG* button to start system configuration.

5.2.1 Configuring the PLL system

The dialog box for the *CONFIG* dialog is shown in Fig. 5.1. On it there are three radiobuttons, labeled *LPLL*, *DPLL*, and *ADPLL*. (See next page, top.) Click one of these radiobuttons to select either an *LPLL* (linear PLL), a *DPLL* (digital

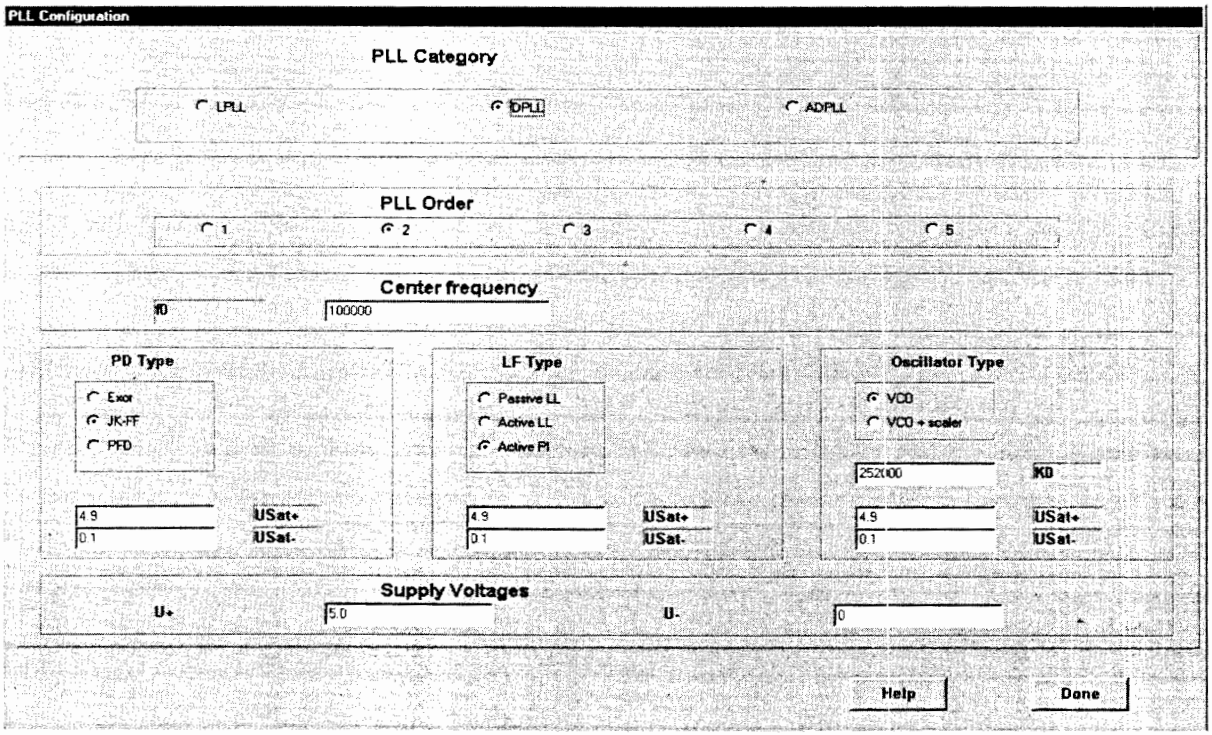
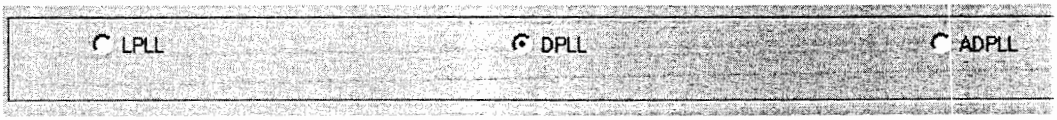


Figure 5.1 Dialog box for PLL configuration.

PLL), or an *ADPLL* (all-digital PLL). As you see, the program starts with a default configuration when it runs the first time. When you close 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.

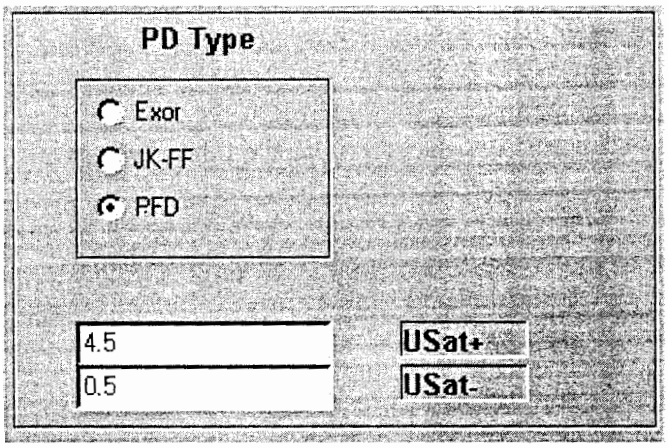


Below these three radiobuttons, there are another five radiobuttons specifying the order of the PLL system under test.

The order number is identical with the number of poles of the system under test. 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, etc. For details, click the *Help* button in the *CONFIG* dialog box.

Below the five radiobuttons there is an edit control specifying the center frequency *f0* of the PLL.

Having selected the center frequency, proceed with entering the parameters of the PLL's building blocks. If you checked the DPLL radiobutton, for example, you would now tell the program which type of phase detector to use:



To do so, click the appropriate radiobutton in the groupbox labeled *PD Type*. For a DPLL system you can specify one of three phase detector types: the EXOR gate, the JK-flipflop, or the phase-frequency detector (PFD). Below the three corresponding radiobuttons there are two edit controls where you can enter the saturation limits of the phase detector circuit under test. Assuming that the PLL is powered from a unipolar 5-V supply, the output of the phase detector cannot reach up to the supply rails for most detector circuits, but stays clamped at same limiting voltage. In the example shown, the upper limit is 4.5 V, the lower limit 0.5 V.

To complete the *CONFIG* specification, enter appropriate values into the other controls of the *CONFIG* dialog box. The last parameters to enter would be the supply voltages, which are defaulted here by 5.0 and 0.0 V, respectively (see Fig. 5.1). To close the box, hit the *Done* button.

Note: The *CONFIG* dialog box is a *modal* one; i.e., you 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 hit, a function is started that checks every entry for validity. First, numbers are tested for correct format. If a user erroneously entered, for instance, "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, etc.

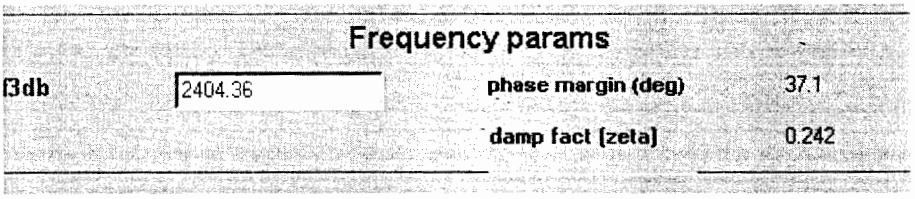
5.2.2 Designing the loop filter

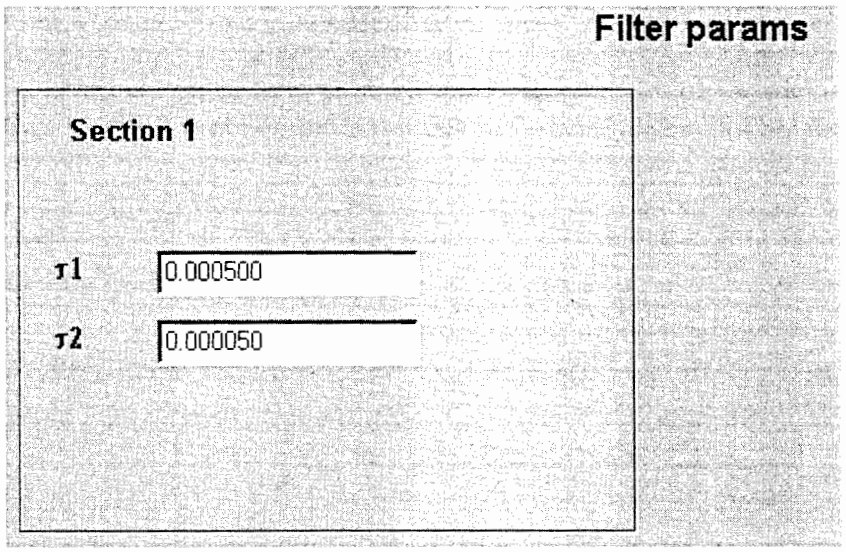
Having closed the *CONFIG* dialog box, we now start the *DESIGN* dialog. Hit the *DESIGN* speedbutton. This displays the *DESIGN* dialog box as shown in Fig. 5.2. On top there is a groupbox labeled *Configuration* showing the setup that has been performed in the *CONFIG* dialog. Below there are another two groupboxes. The upper is named *Frequency params*, the lower *Filter params*. (See next page, top.)

When the *DESIGN* dialog box starts, the filter params (here $\tau 1$ and $\tau 2$) are taken from the values that have previously been entered, or from the default values when the program runs the first time. Moreover, the program computes the frequency params (here *f3db*, *phase margin*, and *damp factor*) on the basis of the given filter params $\tau 1$ and $\tau 2$. *f3db* is the 3-dB corner frequency of the closed-loop gain of the PLL (i.e., 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 *f3db* value into the frequency params box, e.g., 5000 Hz, then hit 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 damping factor and phase margin (see Fig. 5.2).
- 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





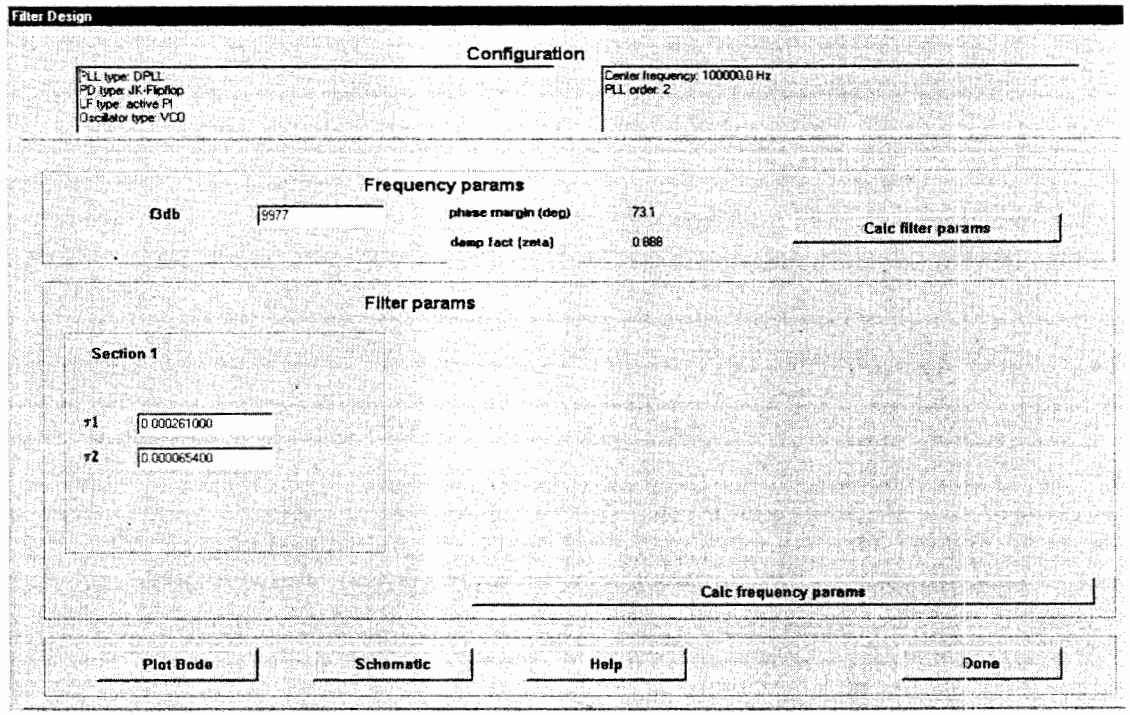


Figure 5.2 Dialog box for loop filter design.

then calculates the resulting values for *f3db*, *phase margin*, and *damp 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.

5.2.3 Analyzing stability of the loop

To analyze stability of the loop, hit the button labeled *Plot Bode*, which is located at the bottom of the *DESIGN* dialog box, Fig. 5.2. This brings up Bode plots for both open-loop and closed-loop phase transfer functions, Fig. 5.3.

This is an example of a fourth-order loop. On screen the curves are displayed in colors. The red curve is the open-loop gain, and the green curve the closed-loop gain. In this text, however, the curves are printed in gray scale. To avoid confusion of open-loop with 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 4 poles and 1 zero, the phase approaches –270° at high frequencies. Here the closed-loop phase curve starts with about 0°.

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°; hence we have a phase margin of roughly 45°. To the right of the curves, three edit windows are displayed. They are labeled *fmin*, *fmax*, and *Pts/Dec*. (See next page, top.) *fmin* is the low end of the frequency scale; *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 example, this is rounded downward

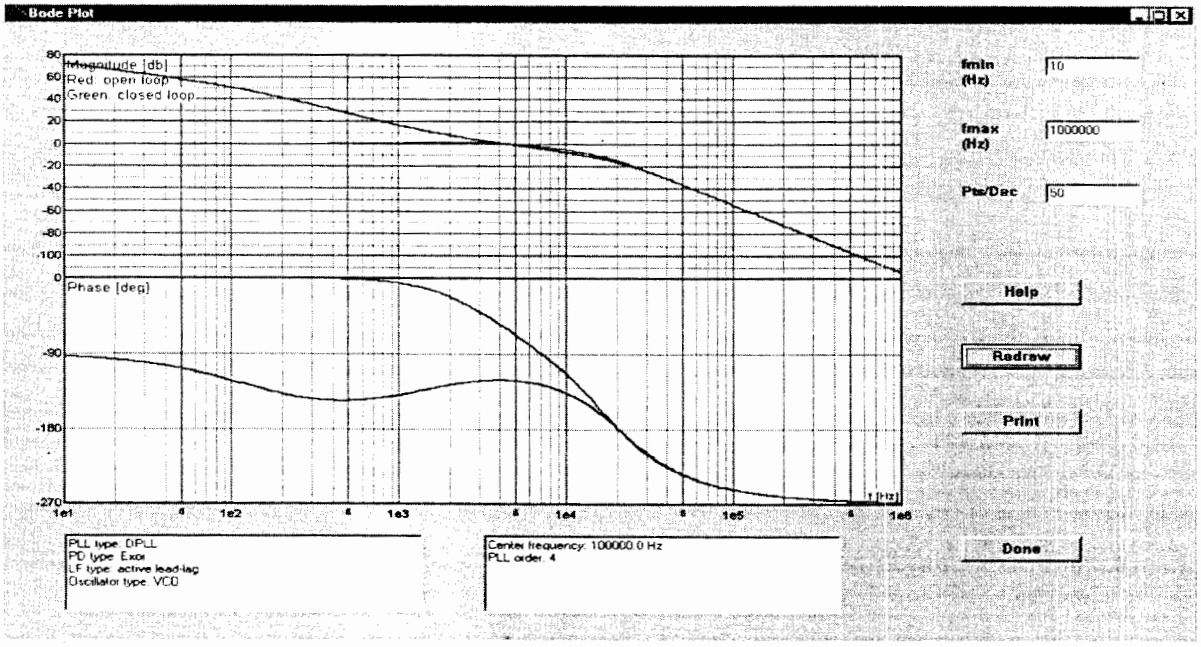
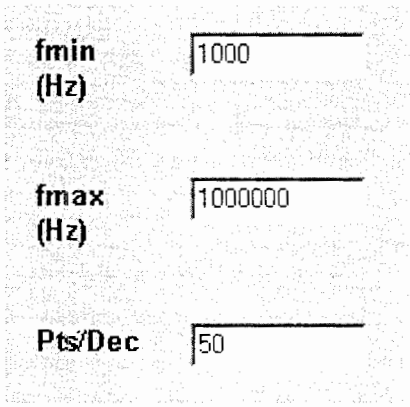
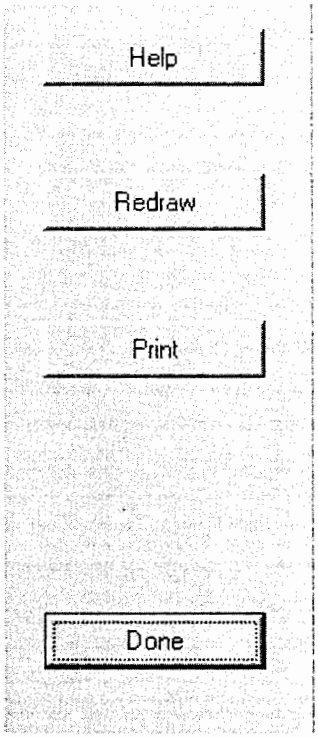


Figure 5.3 Bode plot for open-loop and closed-loop gain of the PLL.

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 there are four pushbuttons in the Bode diagram dialog box:

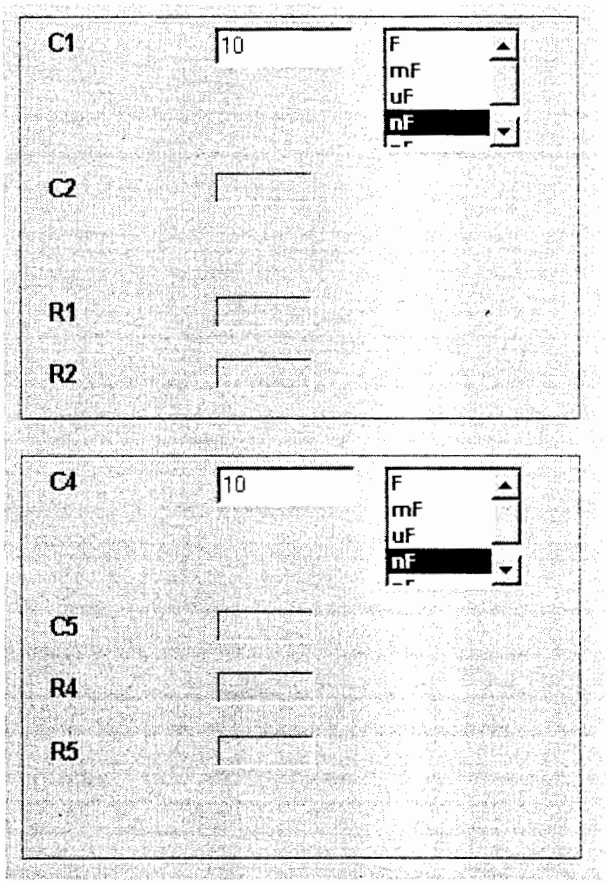


If you have entered different values for *fmin*, *fmax*, or *Pts/Dec*, hit the *Redraw* button to recalculate and replot the new Bode diagram. You can print the Bode plots by hitting the *Print* button. Hitting the *Done* button closes the Bode diagram dialog box.

5.2.4 Getting the loop filter schematic

On bottom of the *DESIGN* dialog box there is a push button labeled *Schematic*. Hitting that button opens the *Schematic* dialog box. The schematic of the loop filter is shown on the left side in Fig. 5.4. This schematic shows a third-order active lead-lag filter.

To the right of the schematic, there are some edit windows and some "static text" windows, which are initially blanked:



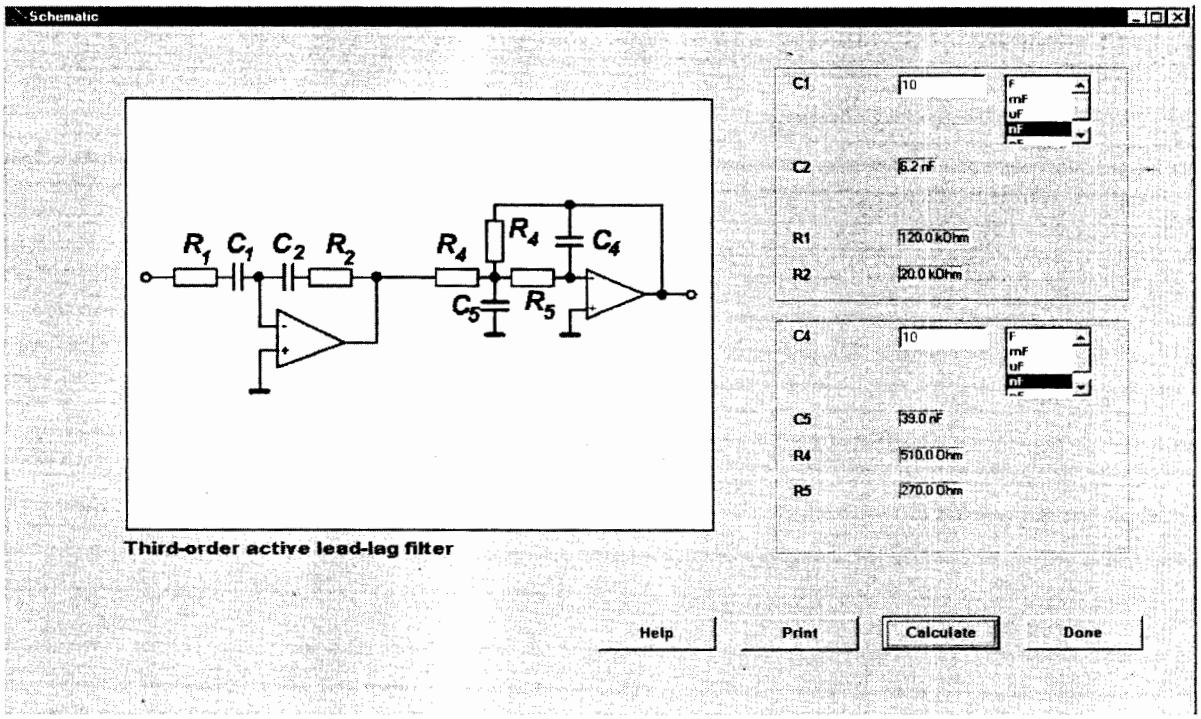
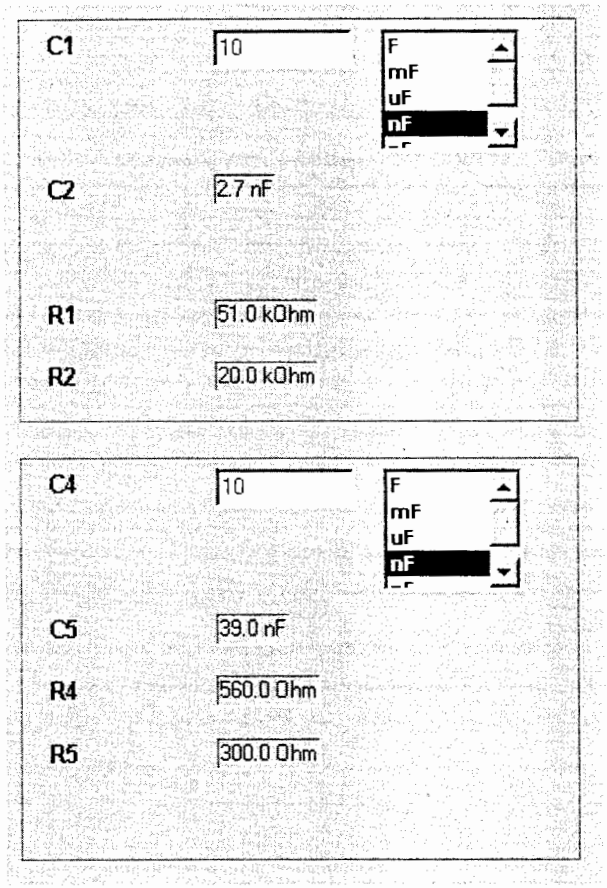


Figure 5.4 Schematic of the loop filter with component values.

In this example, the user can select appropriate values for capacitors C_1 and C_4. There are two items to enter: a value and the unit (F, mF, μF, nF, pF). Hitting the *Calculate* button (Fig. 5.4) starts the computation of all 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, . . . , 8.2, 9.1 Ω, etc.)

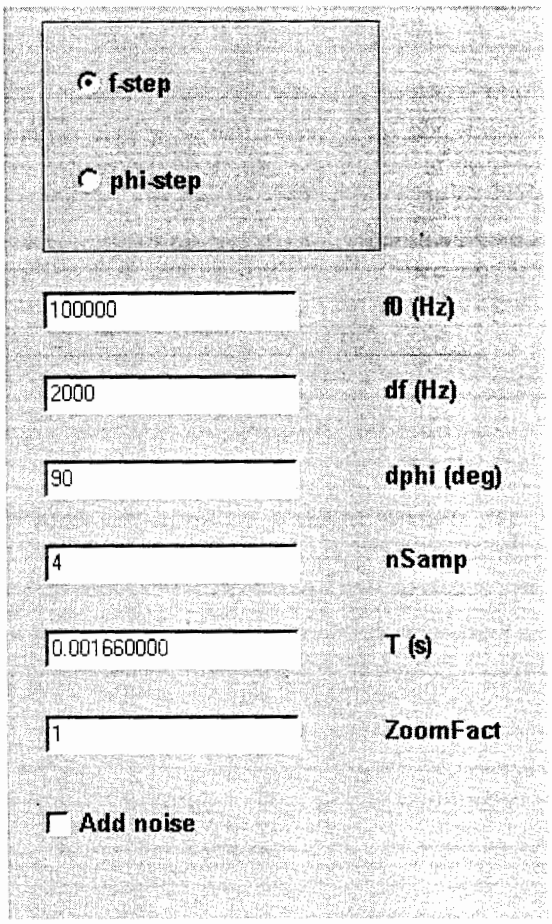
After the *Calculate* button is clicked the display will look something like this:



The values of C_1 and C_4 should be set such that the resulting resistor values have "reasonable values," i.e., in the range of 1 to 100 kΩ. In the example above, the user may wish to get larger values for R_4 and R_5 and would probably enter a lower C_4 (e.g., 1 nF). The schematic, including element values, can be printed out by using the *Print* button (Fig. 5.4).

5.2.5 Running simulations

We are still navigating through the *DESIGN* dialog box. Having completed our design, we can proceed to simulations. To do so, close the design dialog by hitting the *Done* button. Clicking the *SIM* speedbutton enters the simulation dialog box, shown in Fig. 5.5.



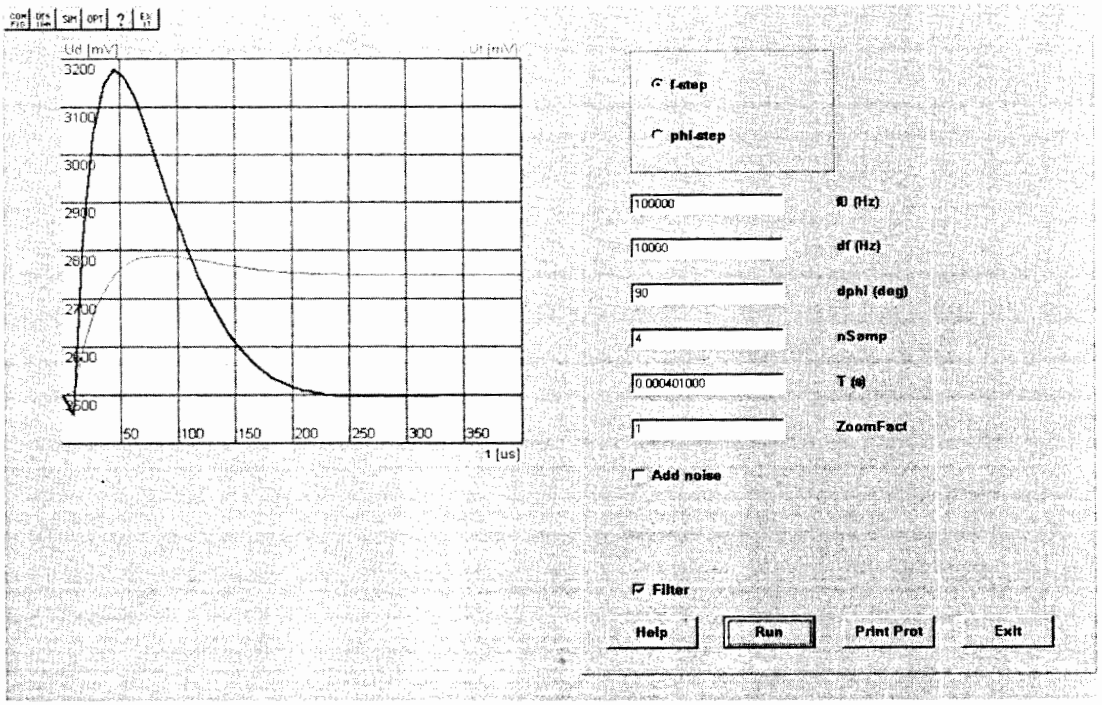


Figure 5.5 Simulation of frequency step applied to the reference input.

Here a number of simulation parameters can be entered. First determine whether a frequency step or a phase step should be applied to the input of the PLL. Check the appropriate radiobutton. In edit control *f0*, the center frequency of the PLL under test must be entered (100,000 Hz in the example). Next, *df* denotes the frequency step applied to the PLL (2000 Hz in the example). The next window specifies the phase step *dphi* (if used). *nSamp* specifies the number of samples to be calculated within one cycle of the PLL simulation; this will be discussed later in this section. *T* is the duration of the simulation, 1.66 ms in the example. Last, a zoom factor (*ZoomFact*) can be specified to zoom the results of the simulation. When *ZoomFact* = 10, for example, the time axis is zoomed by a factor of 10. You can browse through the results by moving the thumb of a scrollbar (as in Fig. 5.13).

When the checkbox *Add noise* is checked, noise is added to the input signal of the PLL under test. This will be explained at the end of this section. To start the simulation, hit the *Run* button. The result of that simulation is shown in Fig. 5.6. This is the response of the PLL to a frequency step $df = 2\,\text{kHz}$ applied to the reference input. The signals u_d and u_f are displayed versus time. On the

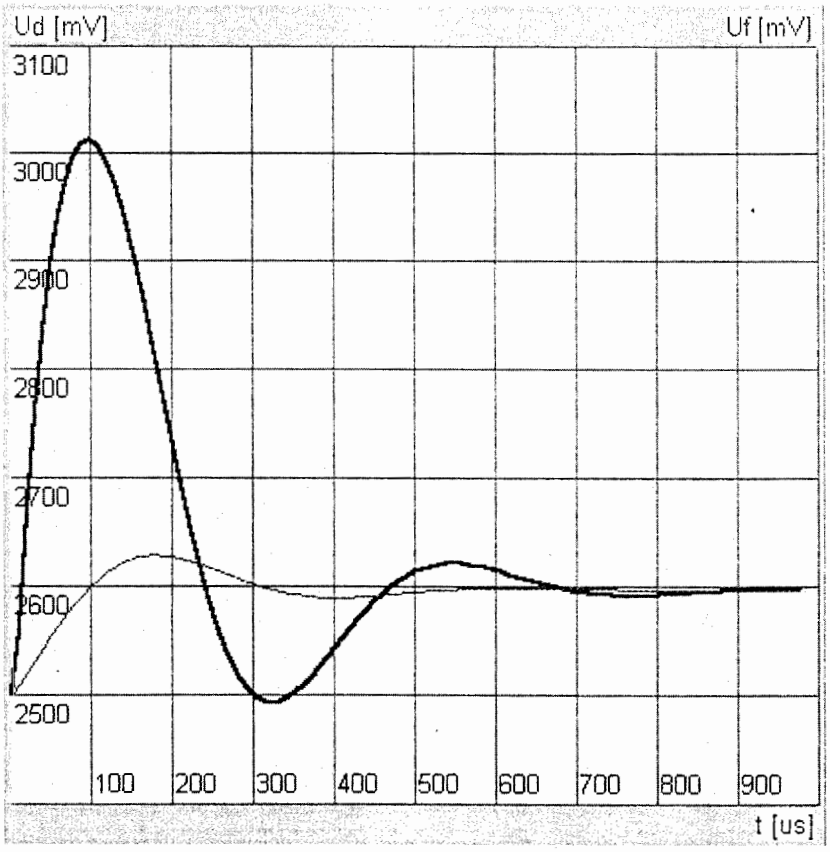


Figure 5.6 Response of the second-order PLL to a frequency step, $df = 2\,\text{kHz}$.

screen the curves are in color; in the book they are printed in gray scale. To indicate which curve represents u_d and which u_f, the u_d signal is plotted with a larger line width. This convention will be used in the following simulations.

The quick tour ends now. To see what happens when you specify different types of phase detectors, loop filters, etc., go back to the *CONFIG* and/or *DESIGN* dialog boxes, make modifications, and return to *SIM* again.

5.2.6 Getting help

The help text in this program is context sensitive. When the speedbutton 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. For example, if the *Help* button in the Bode window is clicked (Fig. 5.3), the corresponding help item is immediately shown.

5.2.7 Shaping the appearance of graphic objects

The user can select a number of options for displaying and printing results of simulations. When the speedbutton labeled *OPT* is clicked, an options window is displayed, Fig. 5.7. At top left there is a list box containing six items. In the example, *AxisTickLabels* is highlighted. For these tick labels the user can select

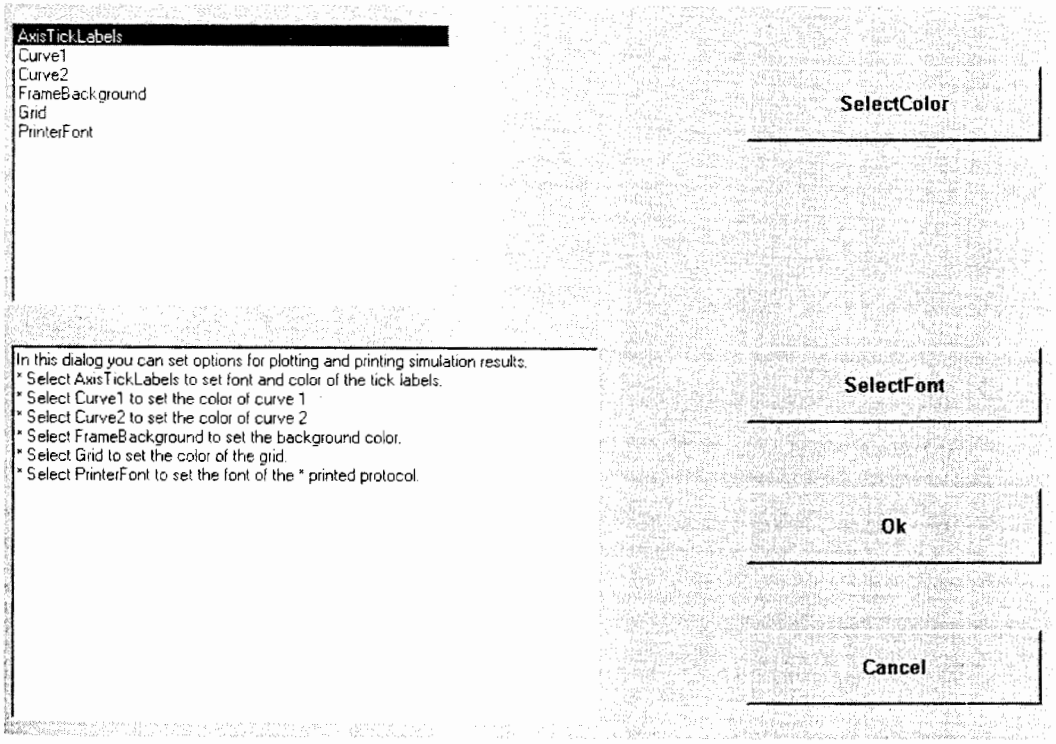


Figure 5.7 Options window.

the color and the font (with all its attributes such as font size, font style, etc.). Option *Curve1* refers to the display of u_d versus time. Here the operator can select line color, line width, etc. All selected parameters are stored in the configuration file when you exit the program. They remain valid when the program is started next.

5.3 Case Study: Design and Simulation of a Second-Order PLL

To see what the program is able to do, we perform a full-fledged PLL design and evaluate its key parameters (lock-in time, lock-out range, etc.) by simulation. Let us start the program and click the *CONFIG* button (Fig. 5.1). In the *CONFIG* dialog box, enter the following parameters:

PLL Category	DPLL
PLL Order	2
Center frequency f0	100,000 (Hz)
PD Type	EXOR
Usat+	4.5 (V)
Usat–	0.5 (V)
LF Type:	Passive LL
Oscillator Type:	VCO (N = 1)
Usat+	4.5 (V)
Usat–	0.5 (V)
Supply Voltages	
U+	5.0 (V)
U–	0

Click the *Done* button to close the *CONFIG* dialog and hit the *DESIGN* speedbutton. We are now in the *DESIGN* dialog box, Fig. 5.2. Let us assume that we want to have a 3-dB bandwidth of 5000 Hz. Thus we enter 5000 in the *f3db* edit window and hit the *Calc filter params* button. The program computes the filter parameters *τ1* and *τ2* required to achieve this goal. We get $\tau_1 = 707\,\mu s$ and $\tau_2 = 126\,\mu s$. At the same time, the resulting damping factor is computed; we get $\zeta = 0.932$. Now we may prefer a lower damping factor; e.g., $\zeta = 0.7$. This can be achieved by entering a smaller τ_2 value; with τ_2 changed to 90 μs the damping factor becomes 0.692. This is OK, but we realize that the 3-dB bandwidth has also decreased; i.e., it is 4415 Hz now. To get the planned value of 5000 Hz, we can make τ_1 smaller, too. After reducing τ_1 to 580 μs we finally get $f_{3dB} = 4977$ Hz and $\zeta = 0.755$, which is almost perfect. We can exit the *DESIGN* dialog now. If you like, have a look on the *Plot Bode* or on the *Schematic*. You also may let the program calculate the R and C values in the schematic, and you may print out your filter design.

Knowing all the parameters of the second-order loop, we first compute the relevant key parameters, just to see how the parameters obtained from the simulations agree with the predicted ones. Using Table 2.2, we get the following results:

Detector gain K_d (from Eq. 2.19)	1.27 V/rad
Natural (radian) frequency ω_n	15,697 s^{-1}
Damping factor ζ	0.754
Lock-in range $\Delta\omega_L$	37,163 s^{-1}
Δf_L	5917 Hz
Lock-in time T_L	400 μs
Pull-in range $\Delta\omega_P$	98,148 s^{-1}
Δf_P	15,628 Hz
Pull-out range $\Delta\omega_{PO}$	54,214 s^{-1}
f_{PO}	8632 Hz

We start simulations by clicking the *SIM* speed button (Fig. 5.5). Knowing that the lock-out range is in the order of 8.6 kHz, we suppose that nothing dramatic happens when we apply frequency steps smaller than about 8 kHz. Let us start with a step $df = 2$ kHz. The result looks like the transient response of a linear second-order system, Fig. 5.8. Again the thicker curve represents u_d, the thinner u_f. After increasing the frequency step df to 8.8 kHz and starting another simulation, we see a dip in the u_d waveform, Fig. 5.9. This represents the situation where the pendulum in the mechanical analogy (Fig. 2.27) made

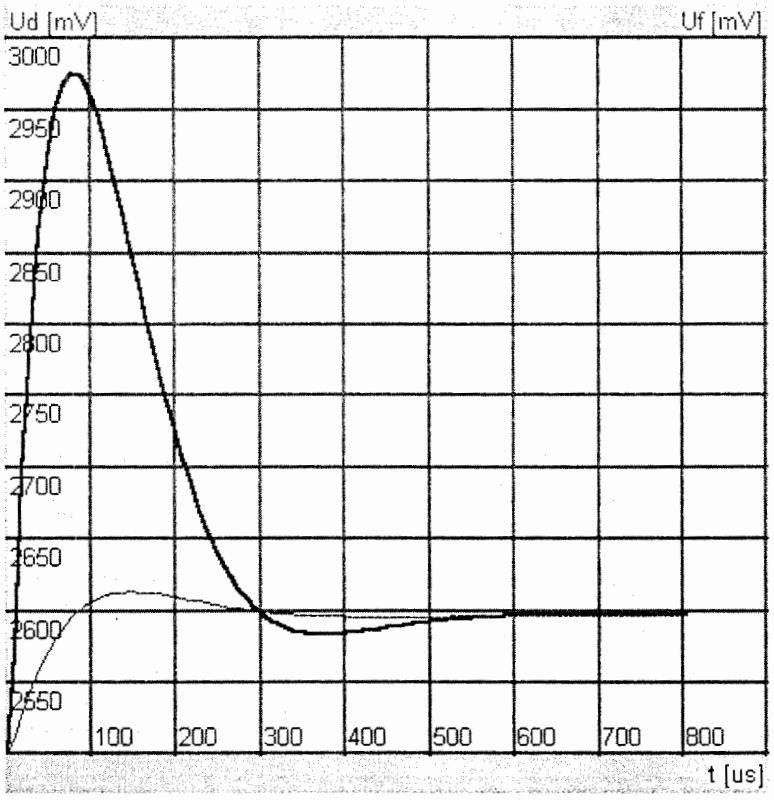


Figure 5.8 Response to frequency step $df = 2$ kHz.

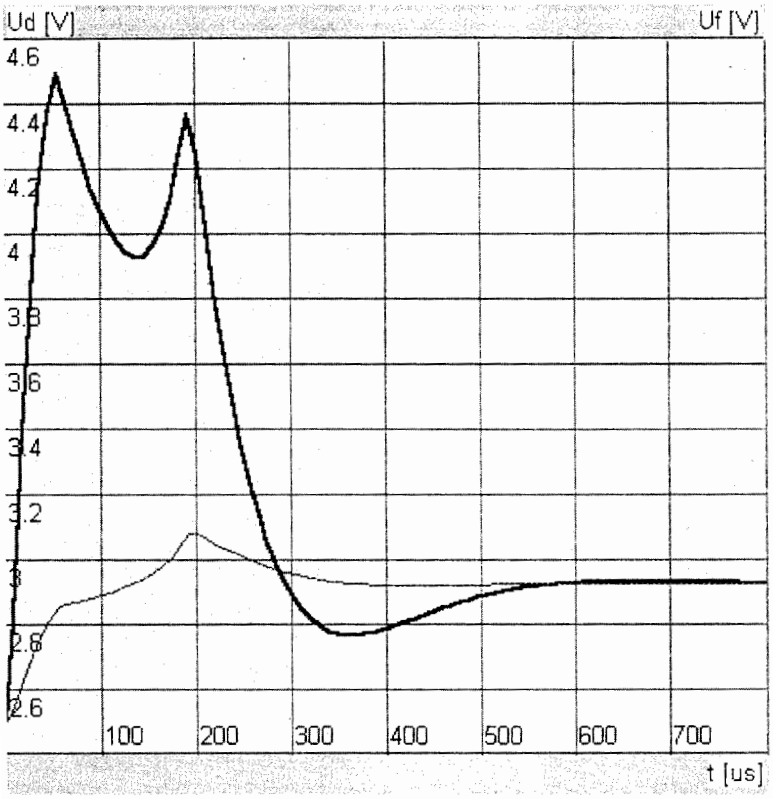


Figure 5.9 Response to frequency step $df = 8.8\,\text{kHz}$ (no lock-out).

a peak deflection greater than 90° but less than 180°. Because the gain of the phase detector becomes smaller for phase errors greater than 90°, the output signal u_d of the phase detector decreases but remains positive. Increasing the frequency step to 8.9 kHz causes the pendulum to tip over now, Fig. 5.10.

The computed value for Δf_{PO} was 8632 Hz, which is quite close to the result of the simulation (8900 Hz). In this simulation the PLL temporarily locked out (the pendulum performed exactly one rotation), but became locked again after a short time. When the frequency step is made even larger (but less than the pull-in range), for example, $df = 14\,\text{kHz}$, the pendulum tips over for a number of revolutions, Fig. 5.11.

The pendulum performed about 11 revolutions before coming to rest. The pumping of the u_f signal is very characteristic, as seen from the thinner curve. After a number of trials we find that the loop no longer pulls in when the frequency step is increased to $df = 17\,\text{kHz}$. The computed value was 15.6 kHz; i.e., the prediction error is about 9 percent.

We did not speak about lock-in range and lock-in time hitherto. It is not easy to measure or simulate the lock-in range. To get the lock-in range we first must force the PLL to unlock by making the difference Δf between (scaled-down) center frequency f_0' and reference frequency f_1 larger than the pull-in range.

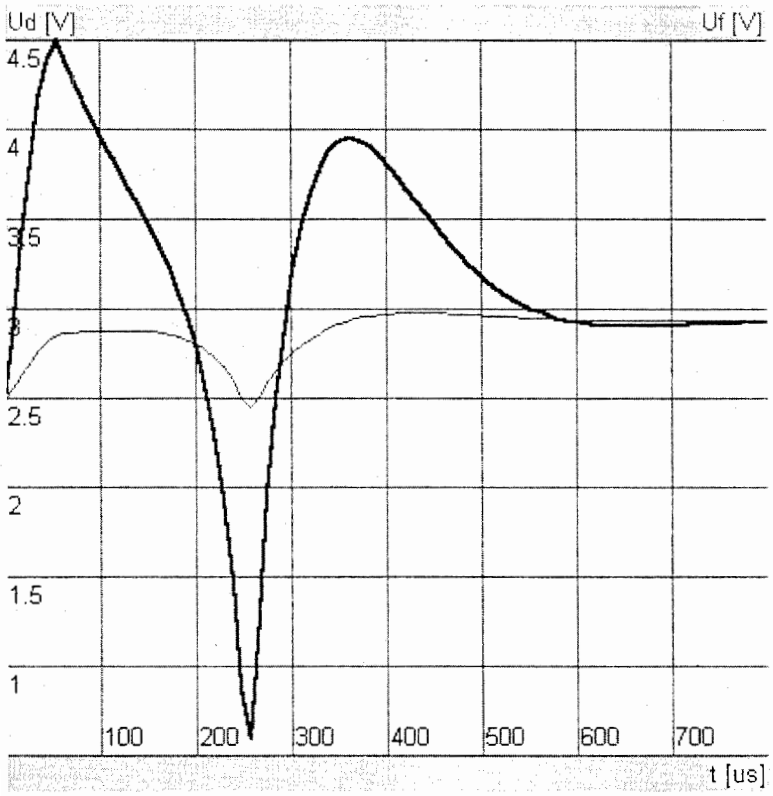


Figure 5.10 Response to step $df = 8.9\,\text{kHz}$ (lock-out).

Then we would suddenly reduce the frequency offset Δf to a value between pull-in range and lock-in range. The loop now reacquires lock. When Δf is larger than the lock-in range, the acquisition process is a slow pull-in process: the pendulum performs a number of revolutions. When Δf becomes less than the pull-in range, however, the acquisition process is a fast lock-in process: the pendulum makes one revolution at most. With this program, it is not possible to simulate this process. The best we can do is a very crude estimation of lock-in range. Look again at the transient response on the 14-kHz step, Fig. 5.11. This frequency step was slightly less than the pull-in range but larger than the pull-out range. Consequently the loop locked out initially. It pulled in with a slow transient (observe the pumping from 0 to 1.3 ms). At the end of this process ($t \approx 1.3\,\text{ms}$) the filter output signal u_f reached a value of about 2.9 V. Beginning at $t = 1.3\,\text{ms}$ the loop locked rapidly, i.e., performed a lock-in process. During that interval of time (about 1.3 to 1.8 ms), u_f was pulled from 2.9 to 3.2 V. A variation in u_f of 1 V corresponds to a frequency variation of $K_0/2\pi \approx 20\,\text{kHz}$. The variation of 0.3 V corresponds to a frequency variation of about 6 kHz. This can be used as a prediction of the lock-in range. The computed value was 5917 Hz.

It must be stated, however, that this method is rather crude and cannot yield precise results. The lock-in time T_L can be estimated quite easily from this kind

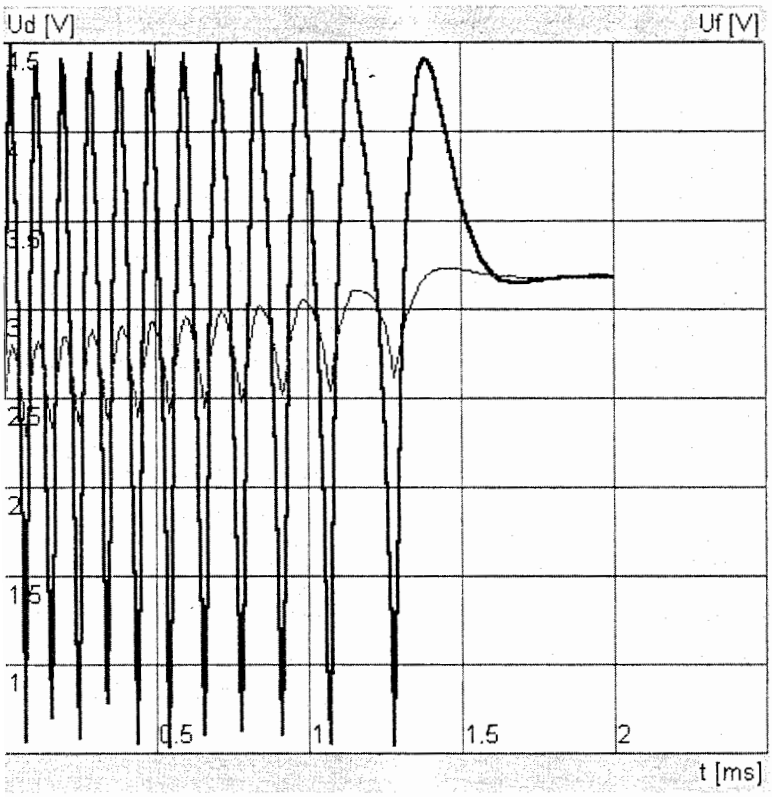


Figure 5.11 Response to step $df = 14\,\text{kHz}$.

of simulation. Theory says that T_L corresponds to roughly one period of the transient response (for $\zeta < 1$). Because the transient response is a damped oscillation whose frequency is the natural frequency, T_L can be estimated from the settling time of the PLL in the case when the frequency step df is less than the pull-out range. From the response of the PLL for $df = 2\,\text{kHz}$, we see that the loop settles in about $400\,\mu\text{s}$, which agrees well with the predicted lock-in time.

You probably became aware 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. Checking again the *SIM* dialog box (Fig. 5.5) we recognize a button labeled *nSamp*, whose default value is 4. *nSamp* denotes number of samples computed in one reference period. This means that the program computes four values for u_d and u_f in each reference period. 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 $2f_1$. To satisfy the sampling theorem we must sample that signal at $4f_1$ at least.

When the checkbox labeled *Filter* (Fig. 5.5) is checked (which is the default) and nSamp = 4, the program computes the *average* of the four values u_d and u_f in every reference period. Doing so, it suppresses the ac signals. 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, setting *nSamp* in the range 4 to 64. We repeat the simulation with *df* = 14 kHz with *nSamp* = 32 and get the result shown in Fig. 5.12.

That looks pretty bad indeed! The curves are smeared over the full window. To see what really happens we will zoom the display. With a zoom factor of 16, the time scale is expanded by a factor of 16. Below the frame window, there is a scrollbar that allows you to scroll through the time window. Now we clearly see what the waveforms really are, Fig. 5.13.

The simulation program also allows you to add noise to the reference signal. To do so check the *Add noise* checkbox, as shown in Fig. 5.14. This opens two

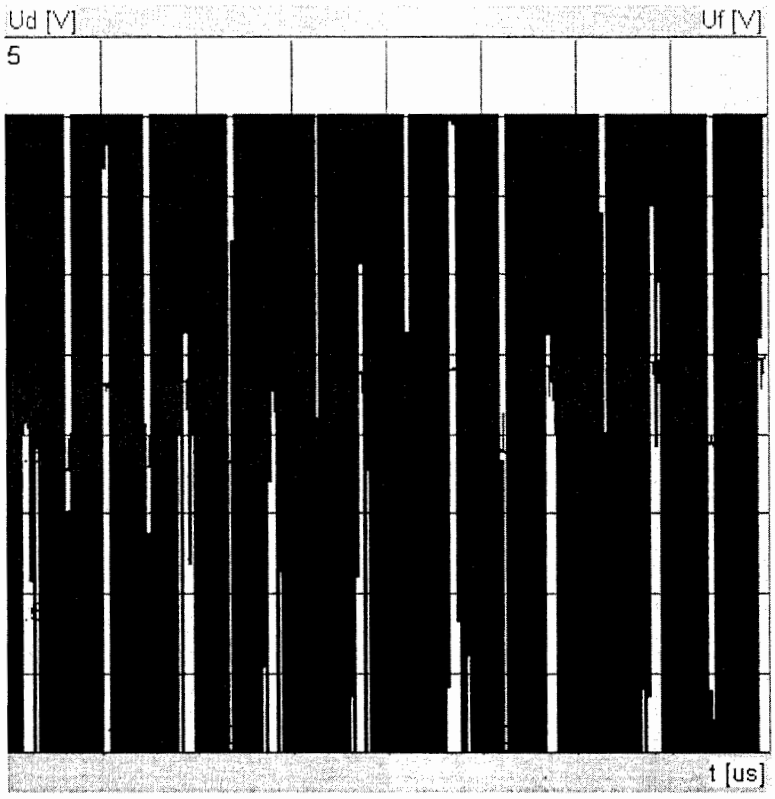


Figure 5.12 Response to *df* = 14 kHz, no filtering, *nSamp* = 32.

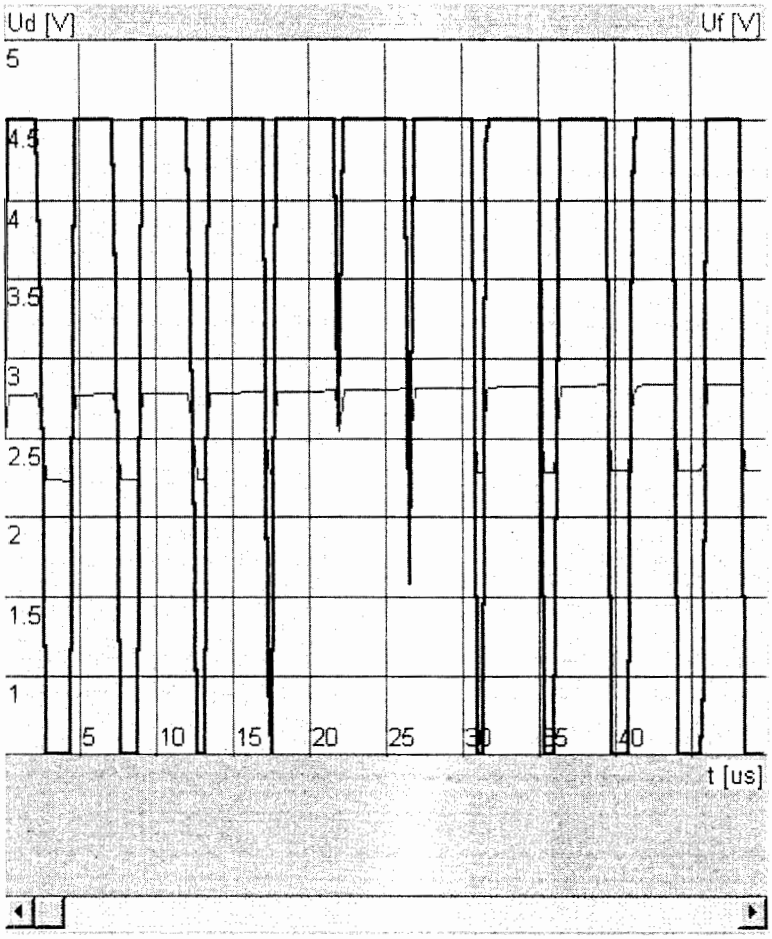


Figure 5.13 Response to df = 14 kHz, no filtering, $nSamp$ = 32, zoom factor = 16.

additional edit controls labeled *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. *NoiseBW (rel)* is set at 0.5. Because the scaled-down center frequency is 100 kHz, the one-sided noise bandwidth is 50 kHz now, which means that there is broadband noise whose power density spectrum reaches from 50 to 150 kHz. The *Help* file contains a lot of additional information on noise. For the parameters entered in this window, the result of the simulation is shown in Fig. 5.15.

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.

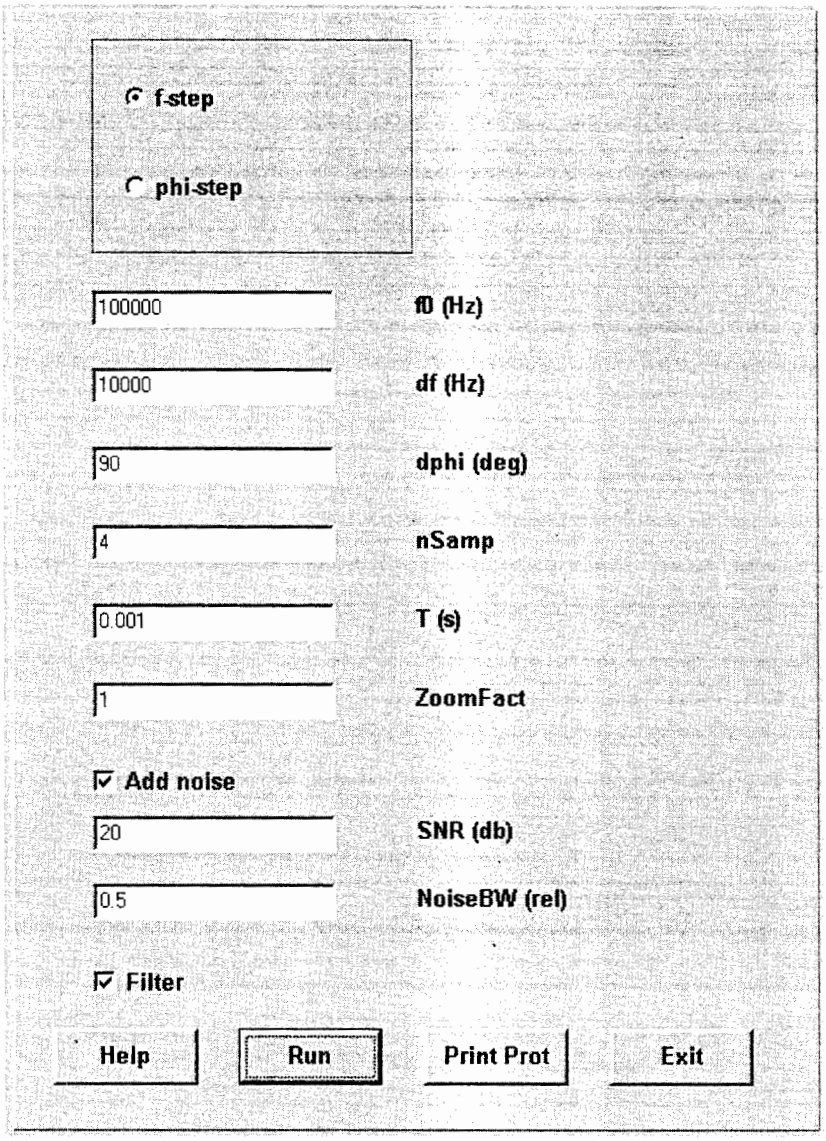


Figure 5.14 Dialog box for PLL simulation, with added noise.

5.4 Suggestions for Other Case Studies

Now that you are familiar with the simulation program, try to simulate other kinds of PLLs. Here 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?

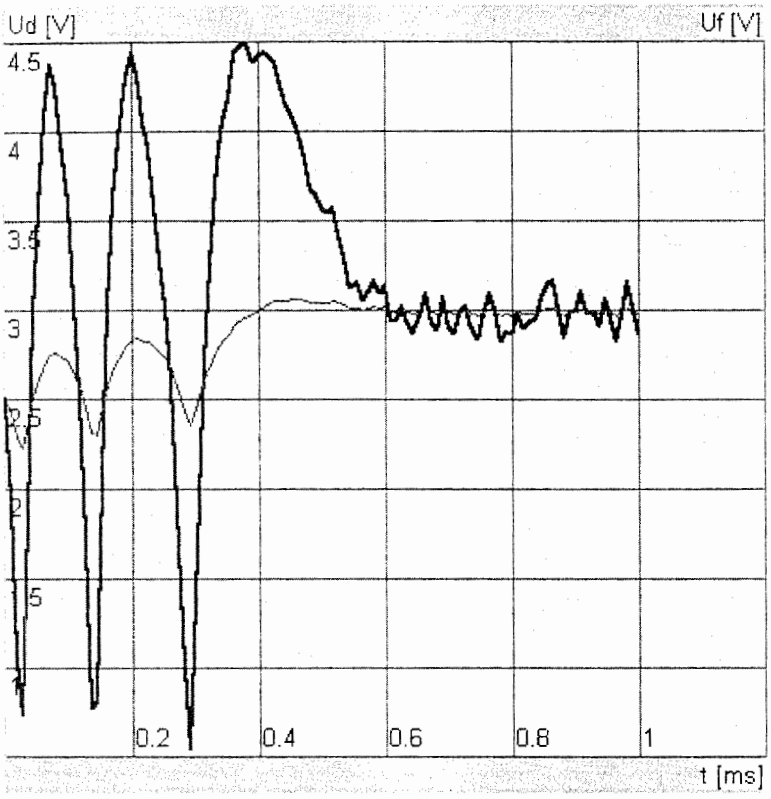


Figure 5.15 Response to $df = 10\,\text{kHz}$, with superimposed noise.

Case study 2: Phase step. Apply a phase step to the reference input. Vary the size of the phase step.

Case study 3: VCO with divide-by-*N* counter. Simulate a PLL whose output frequency is N times larger than the reference frequency. Check how the damping factor ζ 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 ζ, K_0, K_d, K_a, different size of frequency step Δf_1). Compare the results of the simulation with the values predicted by theory; use the formulas in Tables 2.1 to 2.4 to compute pull-in time T_P. Where can you find the largest deviations between theory and practice?

5.5 Displaying Waveforms of Tristate Signals

As we know, the output signal of the PFD is a tristate 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 easily be displayed, but what about the high-Z state? In the sim-

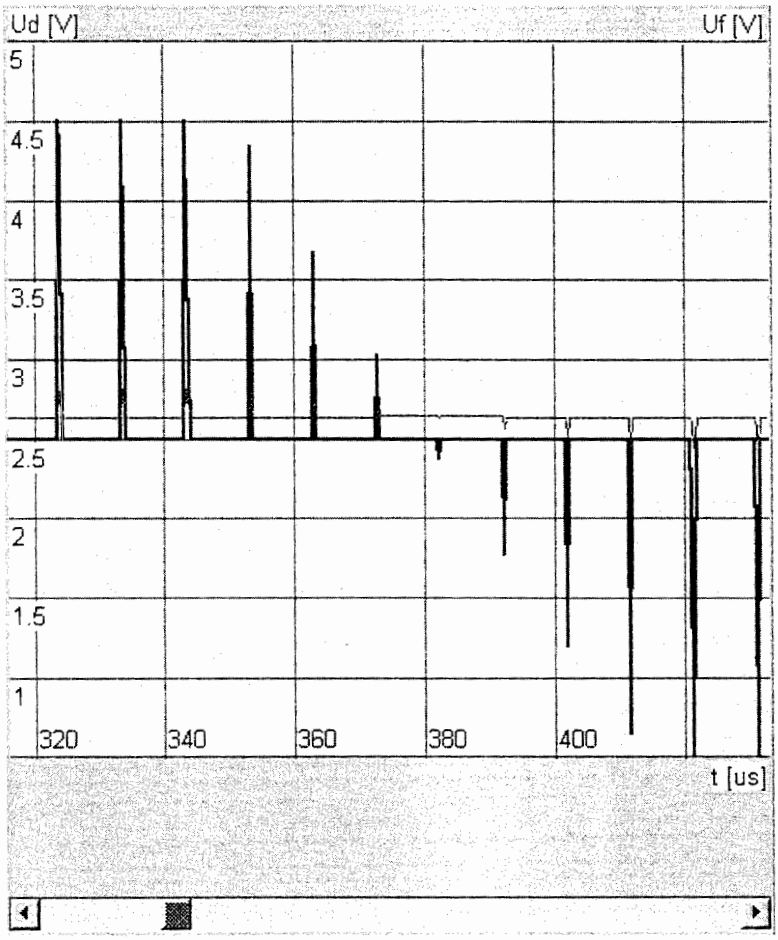


Figure 5.16 Simulation of an acquisition process using the PFD.

ulation program, the high-Z state is represented by a signal level that 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 5.16 shows an example. The time axis is zoomed by a factor 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; at the right it switches to the low state.

Chapter

6

All-Digital PLLs (ADPLLs)

6.1 ADPLL Components

As we have seen in Chap. 2, the classical "digital PLL" (DPLL) is a semianalog 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 does away with these analog-circuitry headaches. In contrast to the older DPLL, it is an entirely *digital* system. Let us 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 as well be a "word" signal, i.e., 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.

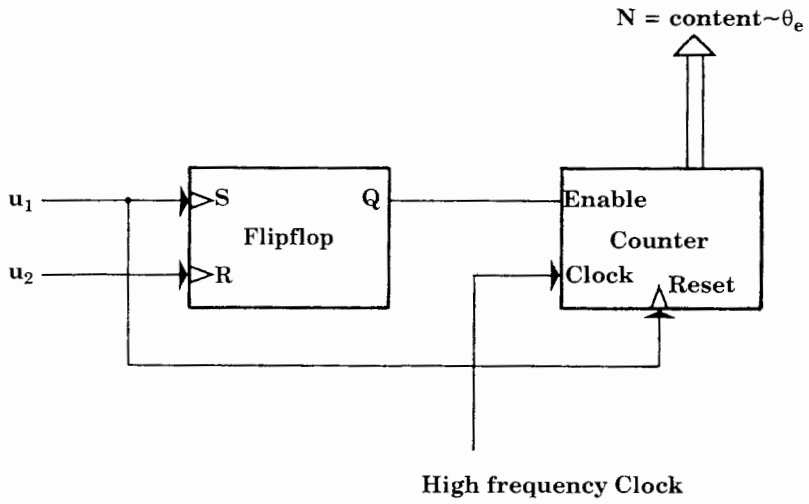
To realize an ADPLL, all function blocks of the system must be implemented by purely digital circuits. Digital versions of the phase detector are already known, but we now have to find digital circuits for the loop filter and for the VCO, too. As we will see in Sec. 6.1.3, the digital counterpart of the VCO is the digital-controlled oscillator (DCO). There are an almost unlimited number of purely digital function blocks for the ADPLL; to save space, we concentrate on the most frequently used.

6.1.1 All-digital phase detectors

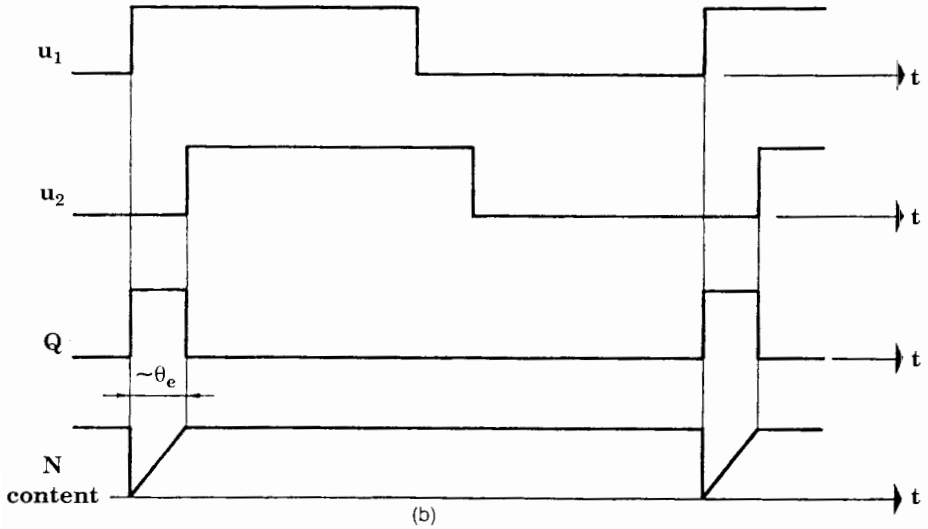
The three most important digital phase detectors have already been discussed in Sec. 2.3.1. When digital word signals instead of bit signals are used, a number of additional phase-detector circuits become available.

A logical evolution of the simple JK-flipflop PD is the FF-counter phase detector illustrated in Fig. 6.1*a*. The corresponding waveforms are shown in Fig. 6.1*b*. 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 θ_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 θ_e, where N is the n-bit output of this type of phase detector. The frequency of the



(a)



(b)

Figure 6.1 Flipflop-counter PD. (*a*) Block diagram. (*b*) Corresponding waveforms.

high-frequency 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).[8] 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. 6.2*a*, and the corresponding waveforms are seen in Fig. 6.2*b*.

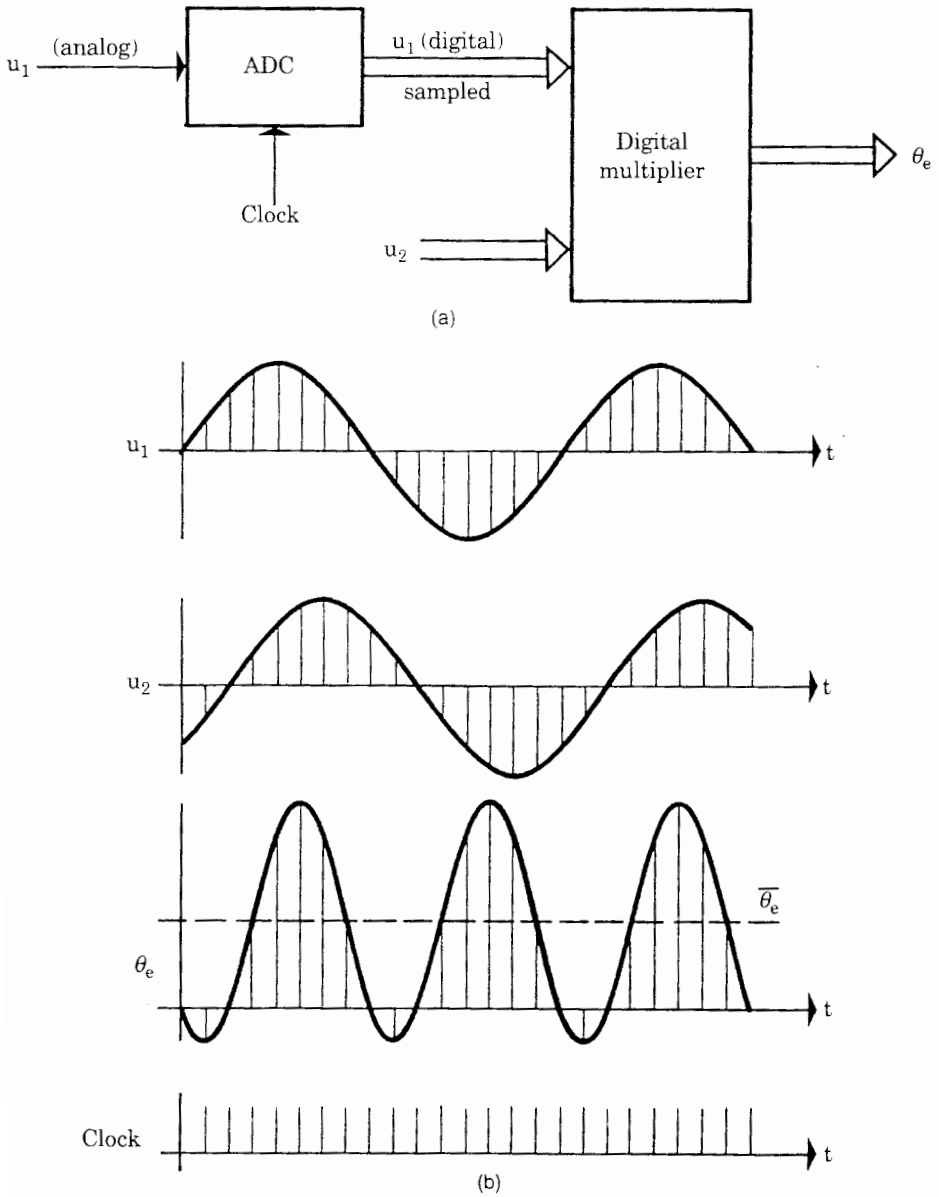


Figure 6.2 Nyquist-rate PD. (*a*) Block diagram. (*b*) Corresponding waveforms.

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 8 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. 6.1.3.) Furthermore, the signal u_2 has been drawn as a sine wave in Fig. 6.2*b*; 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. 2.3.1. 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 θ_e is also shown in Fig. 6.2*b*. Its average value will have to be filtered out by a succeeding digital loop filter, as will be demonstrated in Sec. 6.1.2.

Still another method of measuring the phase error is the *zero-crossing technique*.[8] The simplest zero-crossing phase detector is illustrated in Fig. 6.3*a*; its waveforms are shown in Fig. 6.3*b*. 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 θ_e is a quasi-continuous signal, as shown by the dashed line in Fig. 6.3*b*.

Figure 6.4 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 $-90°$ irrespective of its frequency.[13] Moreover, the gain of the Hilbert transformer is 1 at all frequencies. In the block diagram of Fig. 6.4*a*, 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

$$u_1(t) = \cos(\omega_0 t + \theta_e) \tag{6.1}$$

the output signal of the Hilbert transformer is given by

$$\hat{u}_1(t) = \cos\left(\omega_0 t + \theta_e - \frac{\pi}{2}\right) = \sin(\omega_0 t + \theta_e) \tag{6.2}$$

The Hilbert-transform PD extracts the phase error θ_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

$$\begin{aligned} \cos\theta_e &= Iu_1 + Q\hat{u}_1 \\ \sin\theta_e &= I\hat{u}_1 - Qu_1 \end{aligned} \tag{6.3}$$

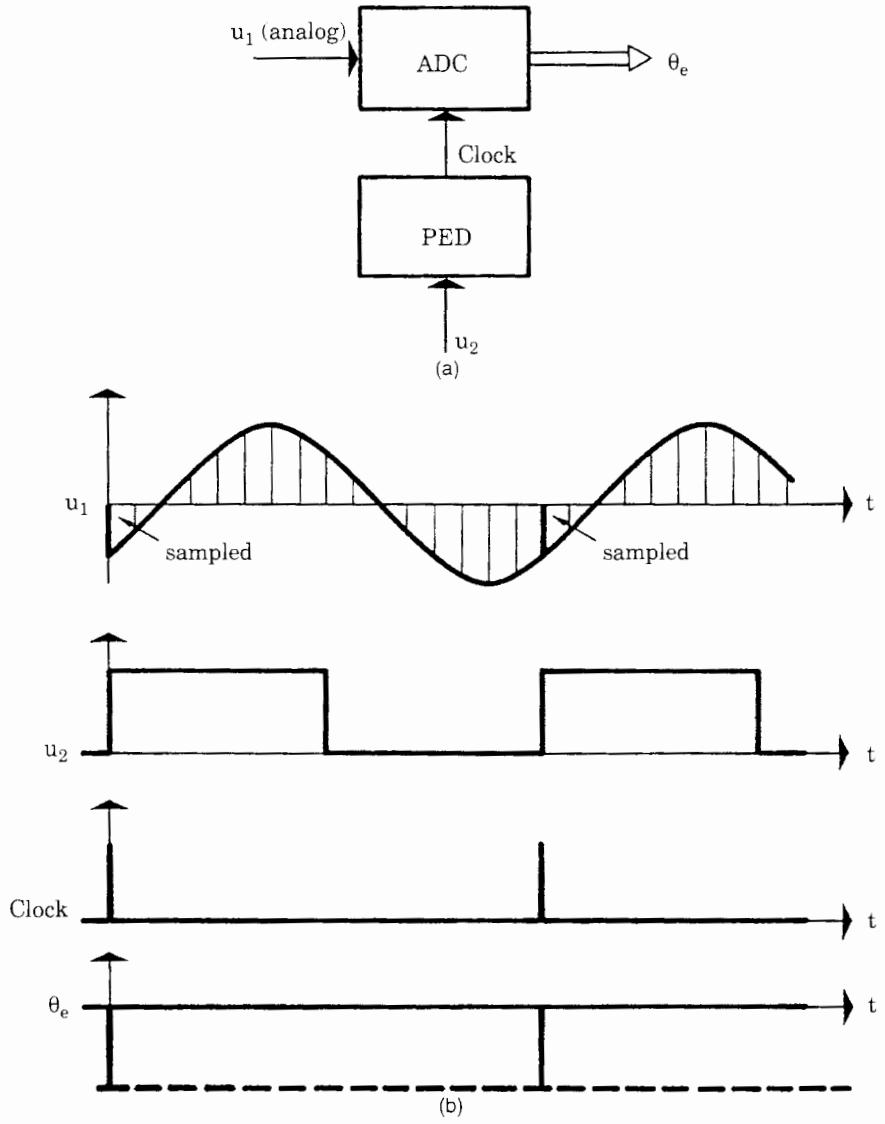


Figure 6.3 Zero-crossing PD sampling the phase error at the positive transitions of the reference signal. (*a*) Block diagram (PED = positive-edge detector). (*b*) Corresponding waveforms.

the sine and cosine of the phase error θ_e are computed. Dividing the sine by the cosine term yields the tangent of phase error; by using a digital algorithm for the inverse tangent ($\tan^{-1}$), the phase error is obtained. The double lines in Fig. 6.4*a* signify that all signals within this device are digital word signals. The signals of the Hilbert-transform PD are shown in Fig. 6.4*b*. 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.

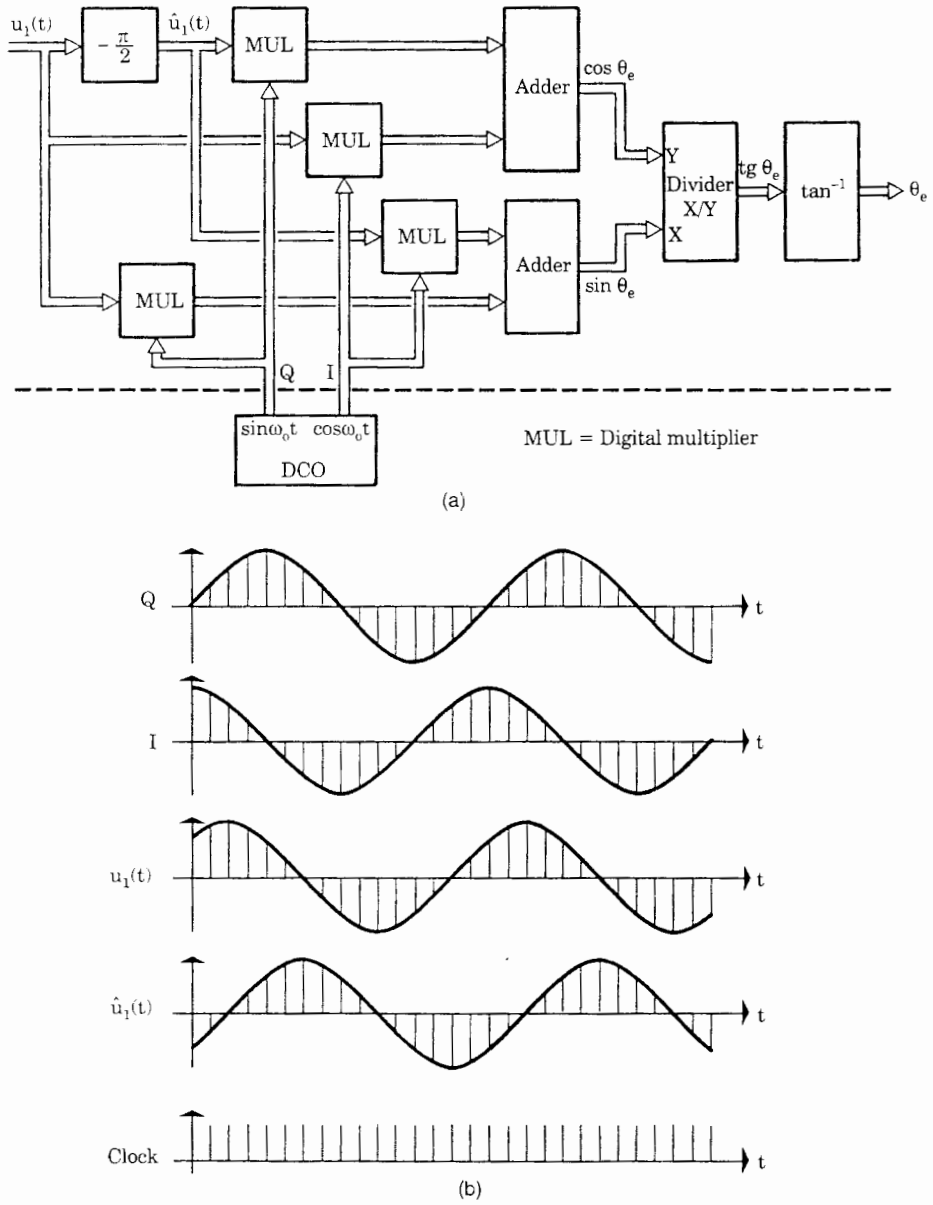


Figure 6.4 Hilbert-transform PD. (*a*) Block diagram. (*b*) Corresponding waveforms.

A similar but simpler way to calculate the phase error is given by the digital-averaging phase detector (Fig. 6.5). 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 $u_r(t)$, but the signals $\cos\theta_e$ and $\sin\theta_e$ are obtained by simply averaging (or integrating)

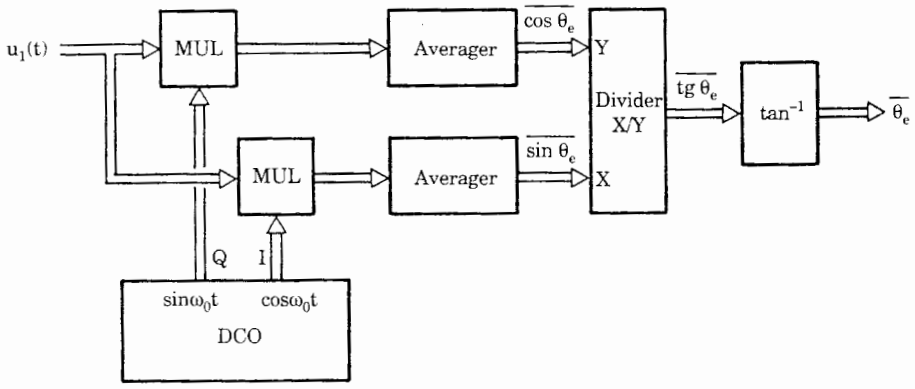


Figure 6.5 Digital-averaging PD.

the output signals of the multipliers over an appropriate period of time.[29] Note that this arrangement already includes a filtering function, defined by the impulse transfer function of the averaging filter used.[14] This method, too, lends itself particularly to implementation by software.

6.1.2 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. 6.1.1 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 tristate signal). It becomes evident that not every all-digital loop filter is compatible with all types of all-digital PDs. We have to 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. 6.6*a*). 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. 6.6*a*, a pulse-forming network is first needed to convert the incoming UP and DN pulses into a counting clock and a direction ($\overline{UP}$/DN) signal (as explained by the waveforms in Fig. 6.6*b*).

On each UP pulse generated by the phase detector, the content N of the up/down counter is incremented by 1. A DN 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, the DN pulses, −1) this filter can roughly be considered an *integrator* having the transfer function

$$H(s) = \frac{1}{sT_i} \tag{6.4}$$

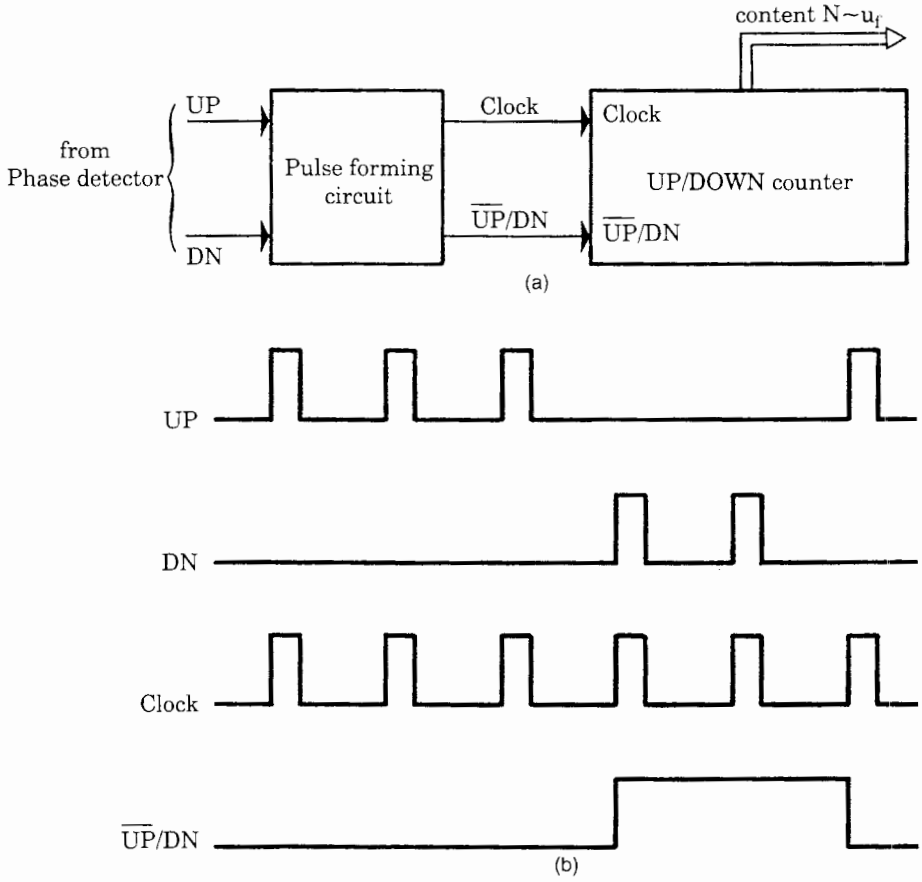


Figure 6.6 Up/down counter loop filter. (*a*) Block diagram. (*b*) Corresponding waveforms.

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 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. 6.7). This loop filter always works together with the EXOR or the JK-flipflop phase detector. As Fig. 6.7*a* 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; i.e., the contents of both counters are 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, ... The operation of the K counter is controlled by the DN/$\overline{\text{UP}}$ signal. If this signal is high,

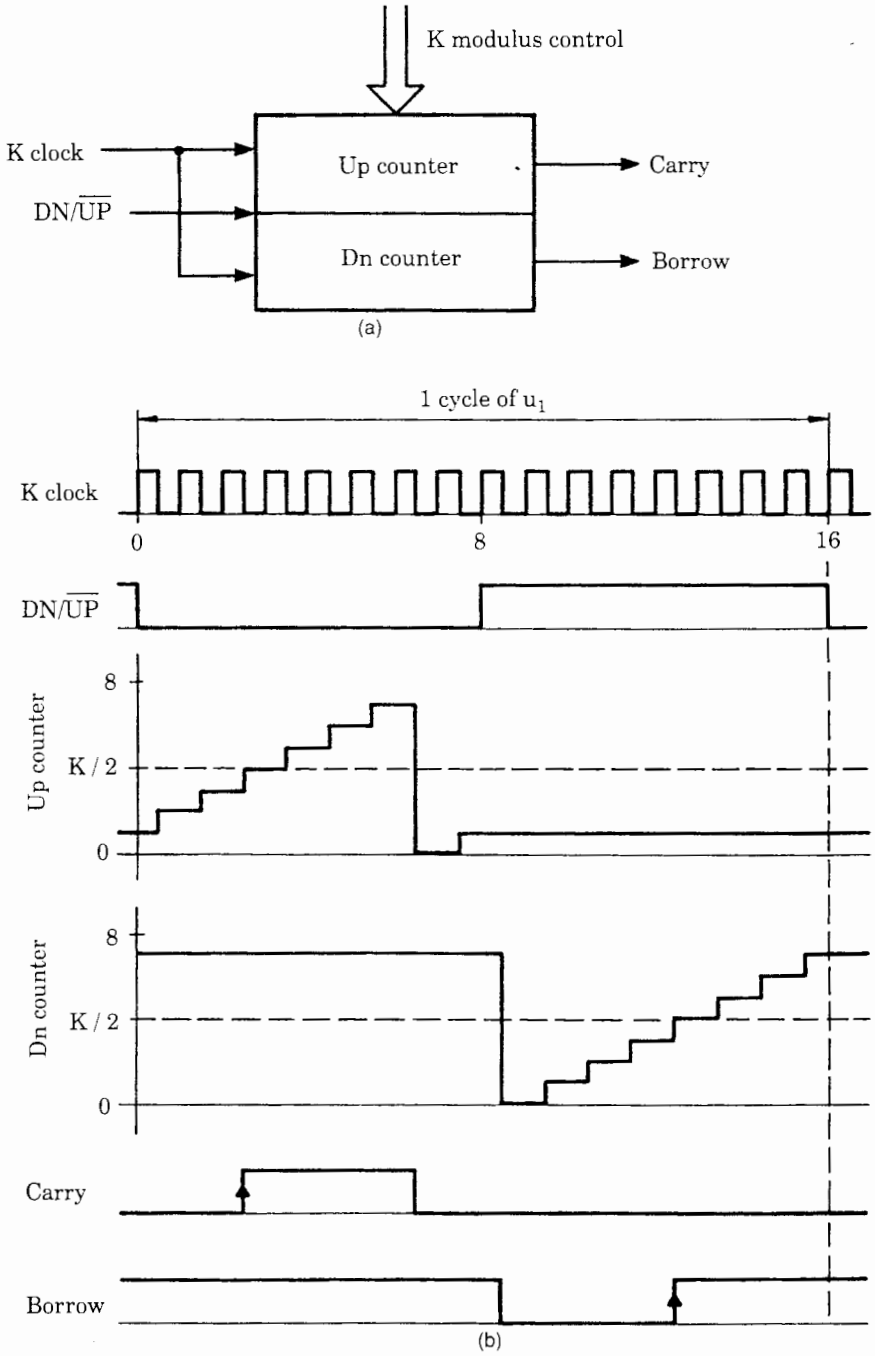


Figure 6.7 *K* counter loop filter. (*a*) Block diagram. (*b*) Corresponding waveforms.

the DN 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 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 gets high when the contents of the DN counter are equal to or more than $K/2$. As will be shown in Sec. 6.3, the positive-going edges of the carry and borrow signals are used to control the frequency of a digitally controlled oscillator.

Figure 6.7*b* shows the signals of the K counter. The DN/$\overline{\text{UP}}$ 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 exactly 50 percent duty cycle. Hence the DN/$\overline{\text{UP}}$ 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. 6.7*b*, we see that the UP counter counts on the first 8 K clock pulses, and the DN counter counts on the next 8 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/$\overline{\text{UP}}$ 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/$\overline{\text{UP}}$ signal is permanently low, the UP counter is active all the time. When the DN$\overline{\text{UP}}$ signal is permanently high, however, the DN counter is working continually. The K counter is part of the popular type 74xx297 ADPLL, which will be treated in Sec. 6.3.

Another digital loop filter is the so-called N-before-M counter (Fig. 6.8). The performance of this filter is very nonlinear. In Fig. 6.8 it is suggested that the 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. 6.8. 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. 6.8, which will output borrow pulses only when

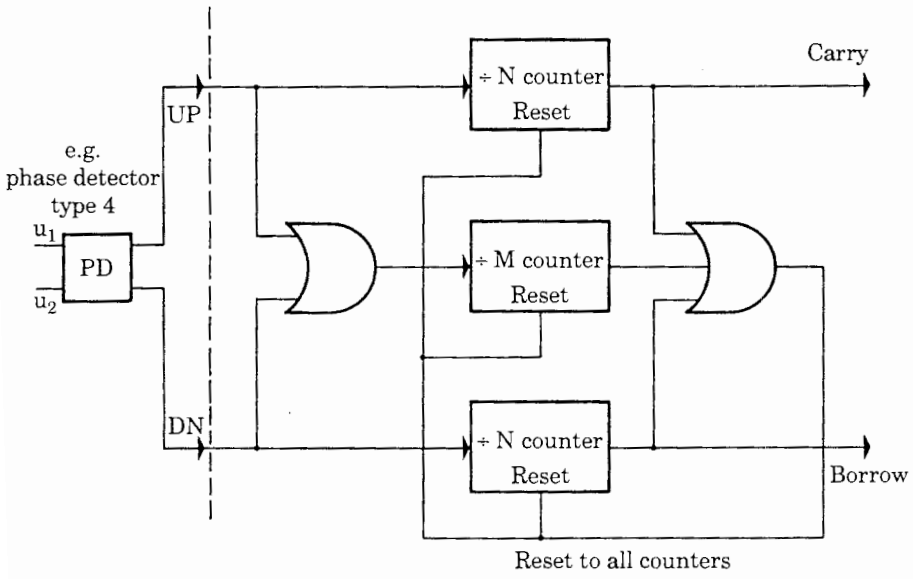


Figure 6.8 Block diagram of the N-before-M loop filter.

a 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 $F(s)$:

$$F(s) = \frac{U_f(s)}{U_d(s)} \tag{6.5}$$

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 the 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; i.e.,

$$F(z) = \frac{U_f(z)}{U_d(z)} \tag{6.6}$$

In implemention of the digital filter, this equation is transformed back into the time domain, which yields a recursion of the form [12,13,14,19]

$$u_f(nT) = b_0u_d(nT) + b_1u_d([n-1]T) + b_2u_d([n-2]T) + \cdots - a_1u_f([n-1]T) - a_2u_f([n-2]T) - \cdots \tag{6.7}$$

The signals u_f and u_d are sampled signals now, which means that they exist only at discrete time instants $t = 0, T, 2T, \ldots, nT$, where T is the sampling interval. The a_i and b_i terms are called *filter coefficients*.

The sampling frequency $f_s = 1/T$ must be chosen much larger than the 3-dB corner frequency of the filter, typically 10 to 20 times the corner frequency.[13.14] The terms $u_f(nT)$, $u_f[(n-1)T], \ldots$ denote the values of the sampled signal u_f at sampling instants $t = nT, (n-1)T, \ldots$. The recursion formula calculates the output signal $u_f(nT)$ in the nth 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.)

6.1.3 Digital-controlled oscillators

A variety of DCOs can be designed; they can be implemented by hardware or by software. We consider the most obvious solutions here.

Probably the simplest solution is the $\div N$ counter DCO (Fig. 6.9). 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. 6.10*a*.[9,16] 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. 6.1.2. The operation of the ID counter follows from the waveforms shown in Fig. 6.10*b*. As Fig. 6.10*a* shows, the ID counter has three inputs, a clock input (ID clock), an increment (INC), and a decrement (DEC) input.

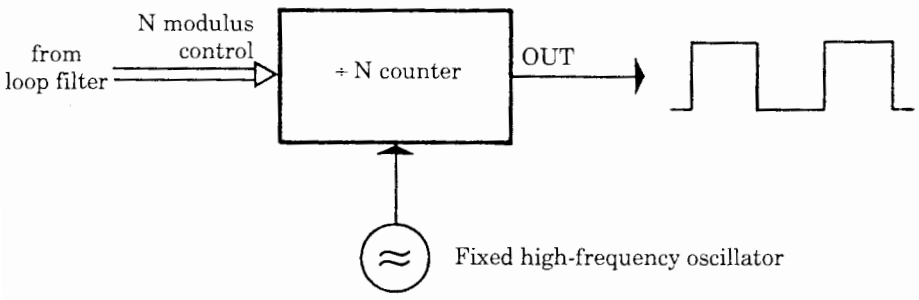
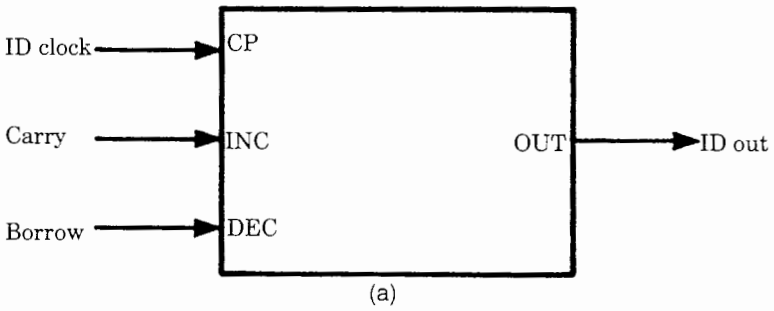
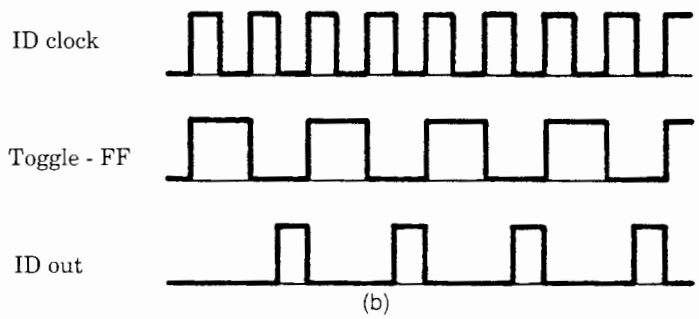


Figure 6.9 Block diagram of a $\div N$ counter DCO.





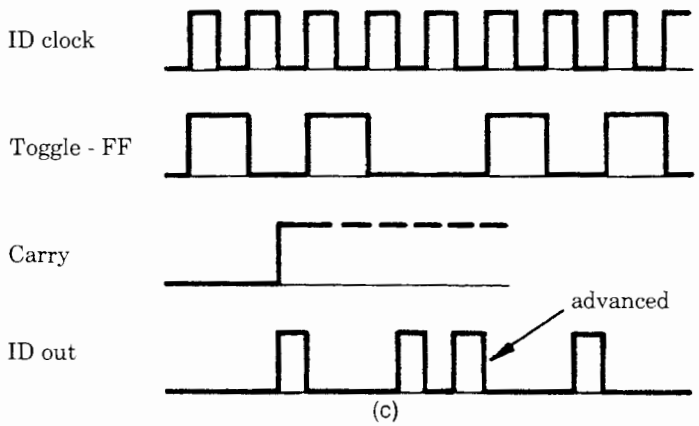


Figure 6.10 ID counter DCO. (*a*) Block diagram. (*b*) Waveforms for the case where no carries and no borrows are applied to the INC and DEC inputs, respectively. (*c*) Waveforms for the case where a carry input is applied when the toggle flipflop is in the 0 state. (*d*) Waveforms for the case where a carry 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.

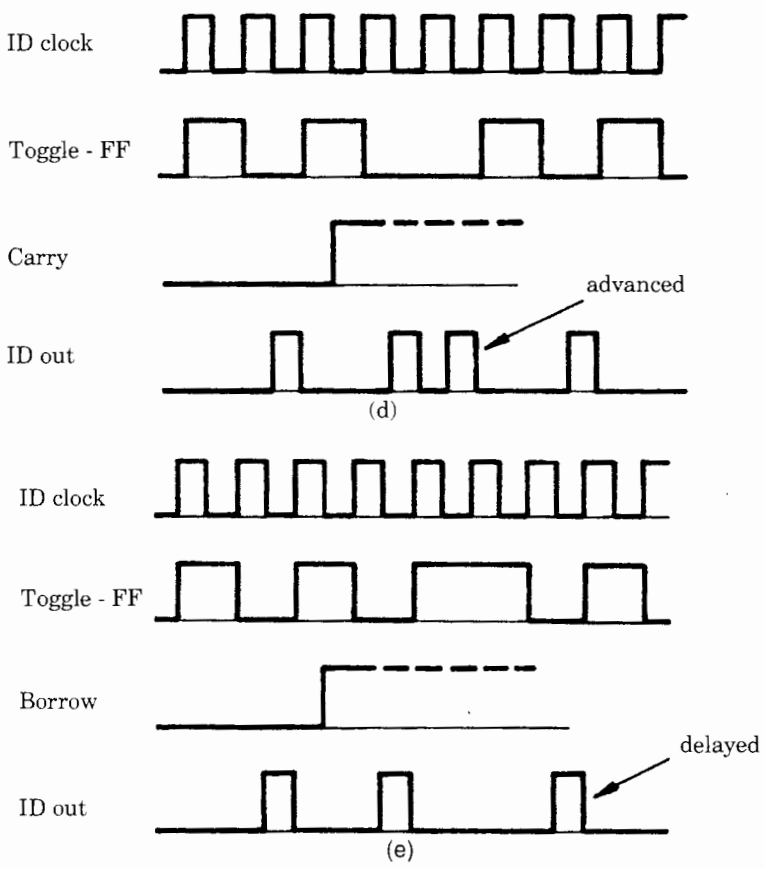


Figure 6.10 (*Continued*)

Carry pulses (as delivered, for example, by a *K* counter, Fig. 6.7) are fed to the INC input 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. 6.10*b*.

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. 6.10*a*. As the waveform *Toggle-FF* in Fig. 6.10*b* 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 $= \overline{\text{IDclock}} \cdot \overline{\text{Toggle} - \text{FF}}$. 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. 6.10*c*), 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 on the next ID clock and remains low during two ID clock periods, as shown in Fig. 6.10*d*. Because the carry can be processed only 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, but at most only two-thirds of that value. This of course limits the hold range of the ADPLL, as will be shown in Sec. 6.3.

Figure 6.10*e* 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 flip-flop 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. 6.3, the limited output frequency range of the ID counter restricts the realizable hold range of the ADPLL. (Note that the explanation of ID counter performance has been slightly simplified; the actual ID counter circuit consists not only 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 $n/2$ hertz when n carries are detected in 1 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 8 carries are detected within 1 second, the "next" IDout pulse is advanced eight times in 1 second by 1/32 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 $\frac{1}{2}$ cycle to be added to the IDout signal, and one borrow pulse causes $\frac{1}{2}$ 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. 8. This type of DCO generates sine and/or cosine waveforms by looking up tables stored in read-only memory (ROM).[30] The block diagram of a waveform-synthesizer DCO is shown in Fig. 6.11*a*.

The waveforms in Fig. 6.11*b* 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 (i.e., it calculates a sample of the synthesized signal at the sampling instants $t = 0, T, 2T, \ldots, nT$, irrespective of the desired frequency). Lower-frequency signals are generated with higher resolution than higher-frequency signals.

In the example of Fig. 6.11*b* it has been arbitrarily assumed that the sampling period is 50 ms; most actual waveform synthesizers operate much faster, 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

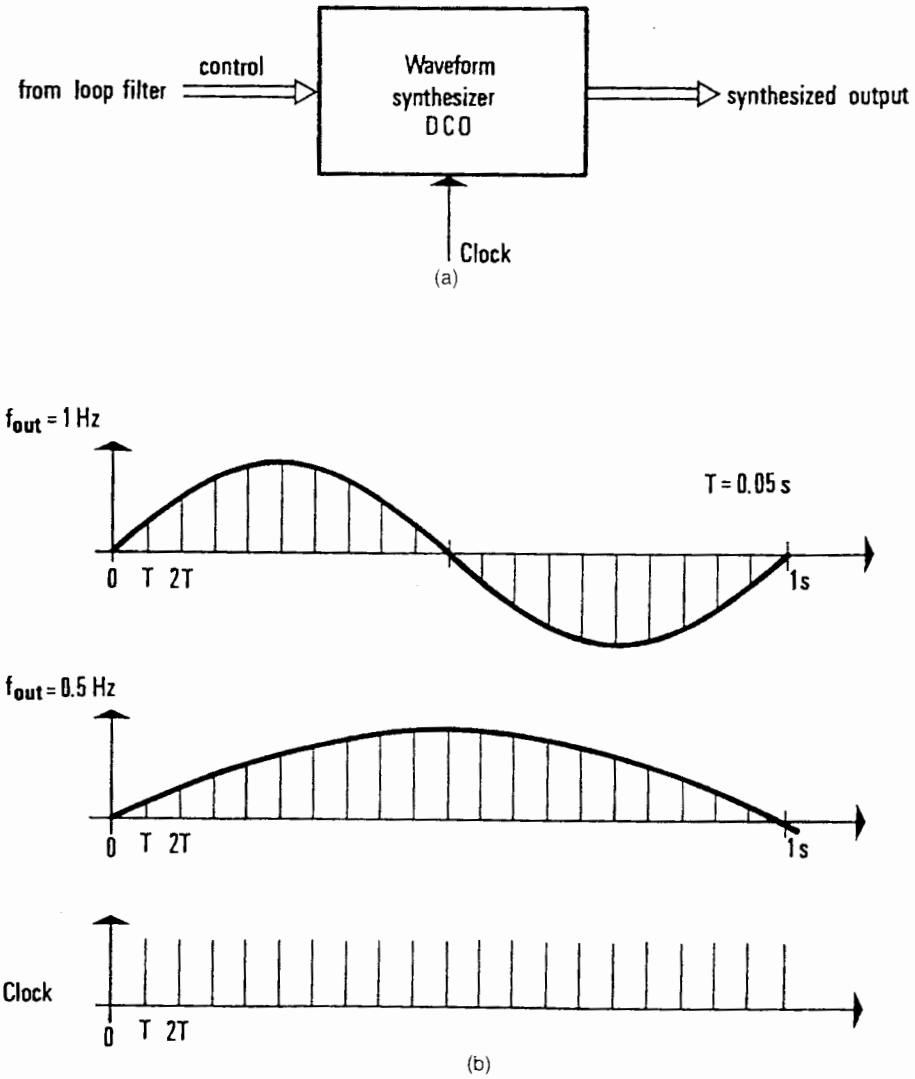


Figure 6.11 Waveform-synthesizer DCO. (*a*) Block diagram. (*b*) The waveforms show how sine waves with frequencies of 1 Hz and 0.5 Hz, respectively, are synthesized.

wave, the synthesizer calculates the sine function for the phase angles $\Phi = 0$ (initial value), $2\pi/20$, $2(2\pi/20)$, $3(2\pi/20)$, . . . , and the phase angle Φ 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

$$\Delta\Phi = 2\pi fT \tag{6.8}$$

where T is the sampling period.

As shown in Fig. 6.11*a*, 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. 6.4*a*).

Digital waveform synthesizers are easily implemented by using single-chip microcomputers.[30] 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. 6.2.

6.2 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 an extended literature on this subject, and a review of the most important systems is 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 to use 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.

6.2.1 ADPLL example 1

The first example of an ADPLL is depicted in Fig. 6.12*a*.[39] In this circuit, the input signal u_1 is first preprocessed by a pulse-forming network (see the dashed box on the left). The generated signals are shown in Fig. 6.12*b*. First, D-flipflop FF1 scales down the frequency of the input signal by a factor of 2. The downscaled 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

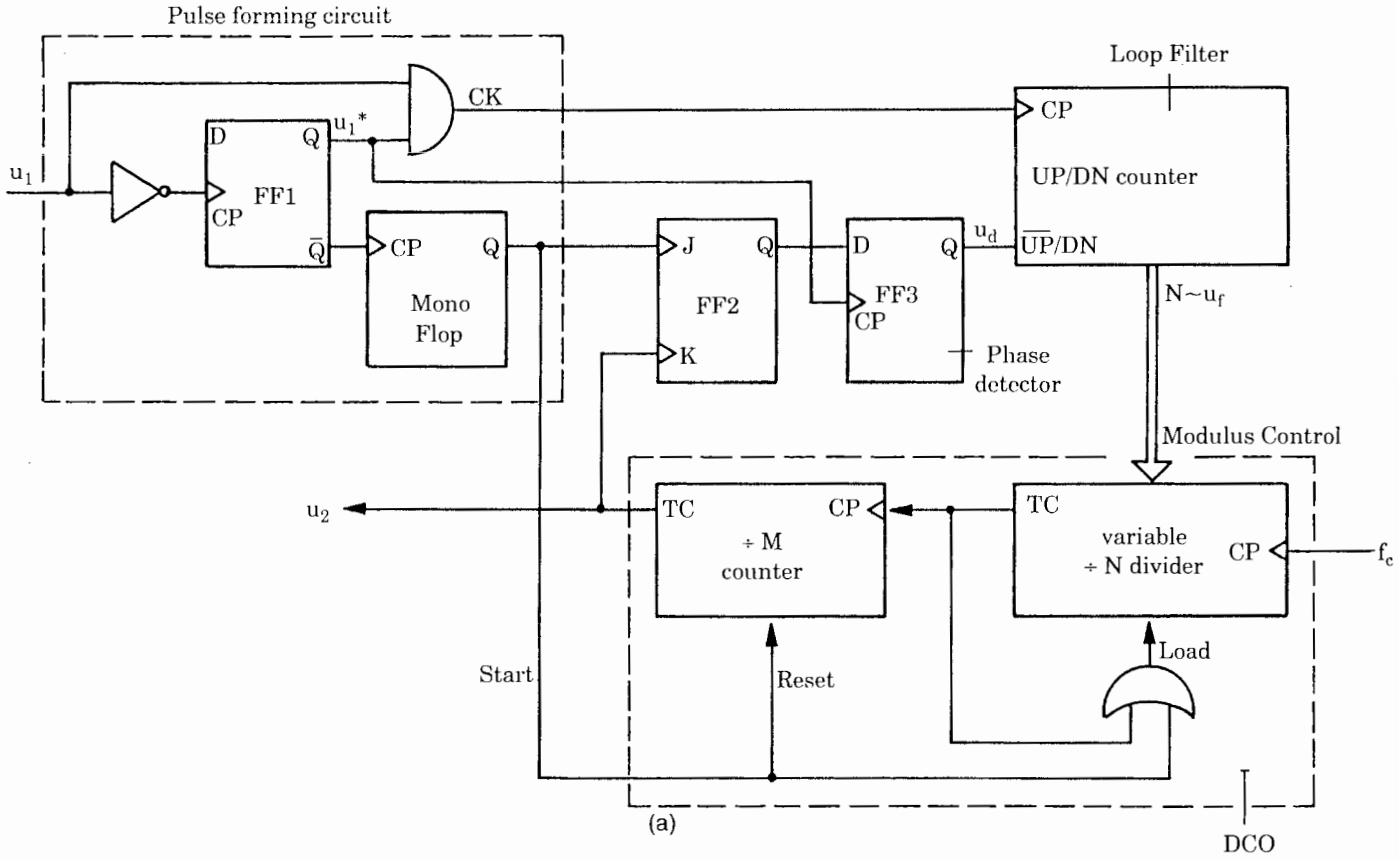


Figure 6.12 All-digital PLL system, example 1. (*a*) Block diagram. (*b*) Corresponding waveforms. Two cases are shown: (1) divider ratio N too small; (2) divider ratio N too large.

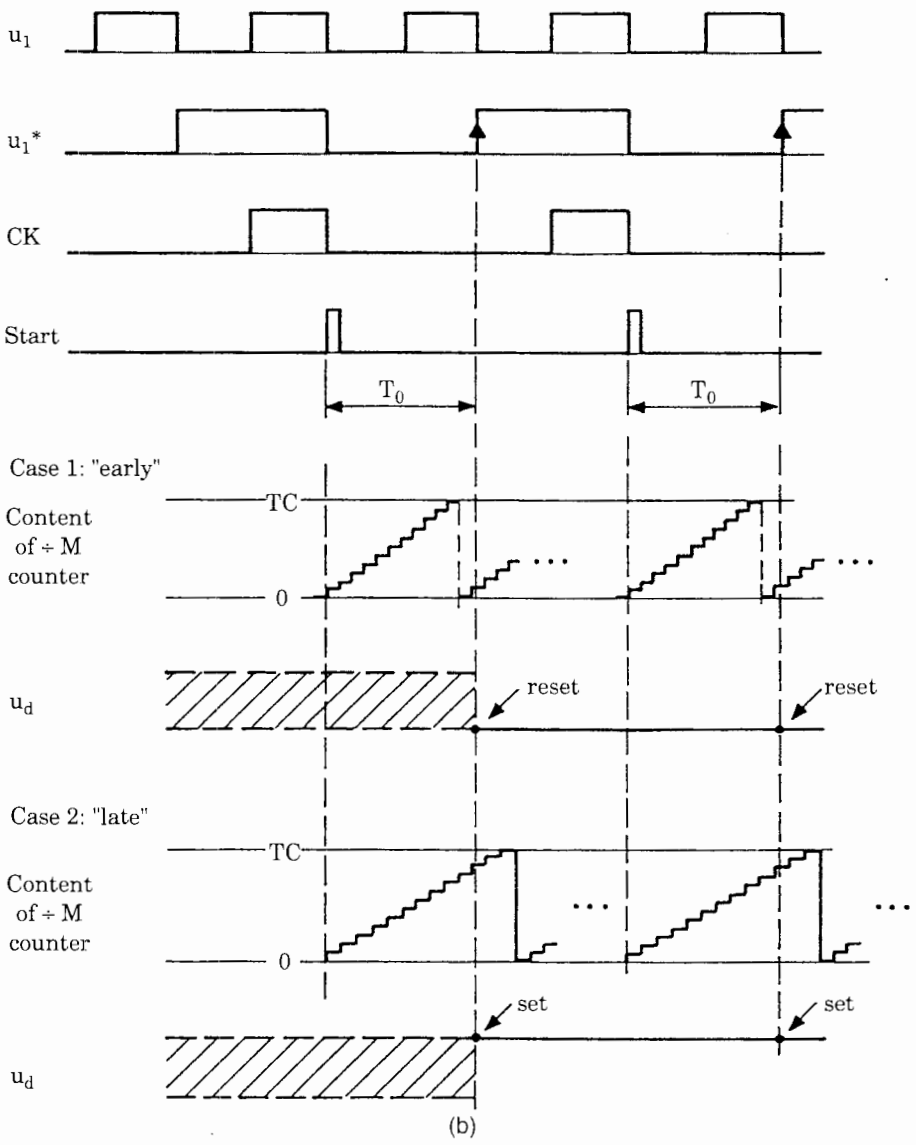


Figure 6.12 (*Continued*)

pulses are labeled *Start*. Their duration must be shorter than the period of the high-frequency clock f_c, which will be discussed in the following paragraph. The dashed enclosure on bottom right of Fig. 6.12*a* 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 on application of 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

$$f_c = NMf_1 \tag{6.9}$$

where f_1 is the frequency of the input signal u_1. If this condition is met, the high-frequency clock delivers exactly NM pulses during one cycle of the input signal u_1, whose period is marked T_0 in Fig. 6.12*b*. 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, shown on right top of Fig. 6.12*a*, 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. 6.12*b*.

The *Start* pulse resets the modulo-M counter on each high-to-low transition of signal u_1* (see the waveform "Content of ÷M counter"). As mentioned above, 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 *Case 1: early* it is 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, i.e., 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. Consequently, the next positive edge of u_1* resets D-flipflop FF3, whose output is labeled u_d (phase detector output) in Figs. 6.12*a* and 6.12*b*. Note that the state of u_d has been indeterminate at the start of the locking process; hence its waveform is drawn by the shaded area in Fig. 6.12*b*. 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 *Case 2: late, N* is too large initially so that the modulo-M counter requires a period longer than T_0 to reach TC. Under this condition, D-flipflop FF3 will

be set 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 NM = 990.1. Supposing that M = 100 (i.e., 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 be 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, which acts as a loop filter, also behaves like an integrator. If a phase error persists for an extended period of time, the content of the UP/DN counter ramps up or down depending on the sign of the phase error, the counter thereby 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 θ_2 is proportional to the integral of applied control signal u_f [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 zeros in this system, it would get unstable. Most happily, this does not occur, because the contents of the counters within the DCO are reset on every *Start* pulse, as described above. In other words, the "integral" term at the output of the DCO is not allowed to wind up. Because of 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.

6.2.2 ADPLL example 2

The second ADPLL system described here is shown in Fig. 6.13. This is the most often used ADPLL configuration; it is available as an integrated circuit with the designation 74xx297, where xx stands for the family specification (HC, HCT, LS, S, etc.). We will analyze this ADPLL in greater detail in Sec. 6.3. The IC contains two phase detectors: an EXOR gate and a JK-flipflop. In the schematic of Fig. 6.13, the EXOR is used. The loop filter is formed by the previously discussed K counter (Fig. 6.7), and the already known ID counter (Fig. 6.10) is used as DCO. This ADPLL system requires an external divide-by-N counter.

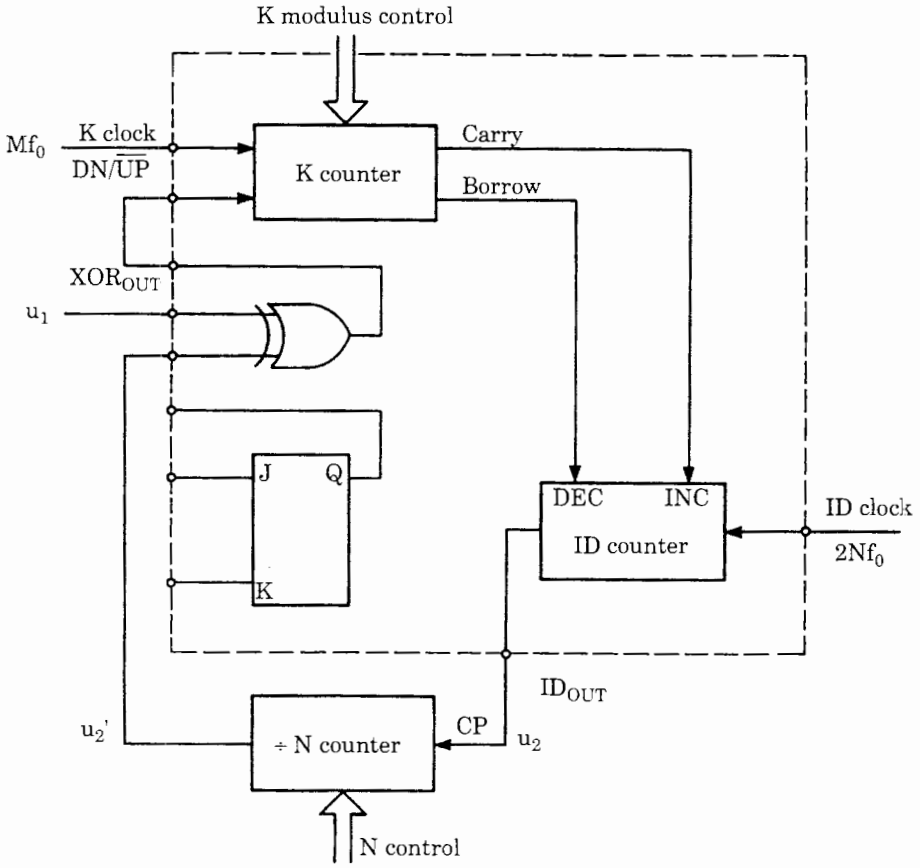


Figure 6.13 All-digital PLL, example 2. This circuit is based on the familiar IC of type 74HC/HCT297. The EXOR phase detector is used here. For the external ÷*N* counter, an IC of the type 74HC/HCT4040 can be used.

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°. In this case, the output signal of the EXOR gate is a symmmetrical 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.

6.2.3 ADPLL example 3

The last example of an ADPLL is illustrated in Fig. 6.14. A similar system has been implemented by software on a TMS320 single-chip microcomputer (Texas Instruments).[30] The system of Fig. 6.14 is built from functional blocks introduced previously:

1. A Hilbert-transform phase detector (Fig. 6.4)
2. A first-order digital loop filter
3. A waveform-synthesizer DCO (Fig. 6.11)

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, $u_d \sim \theta_e$. The output signal u_d is digitally filtered by the loop filter, which performs the operation

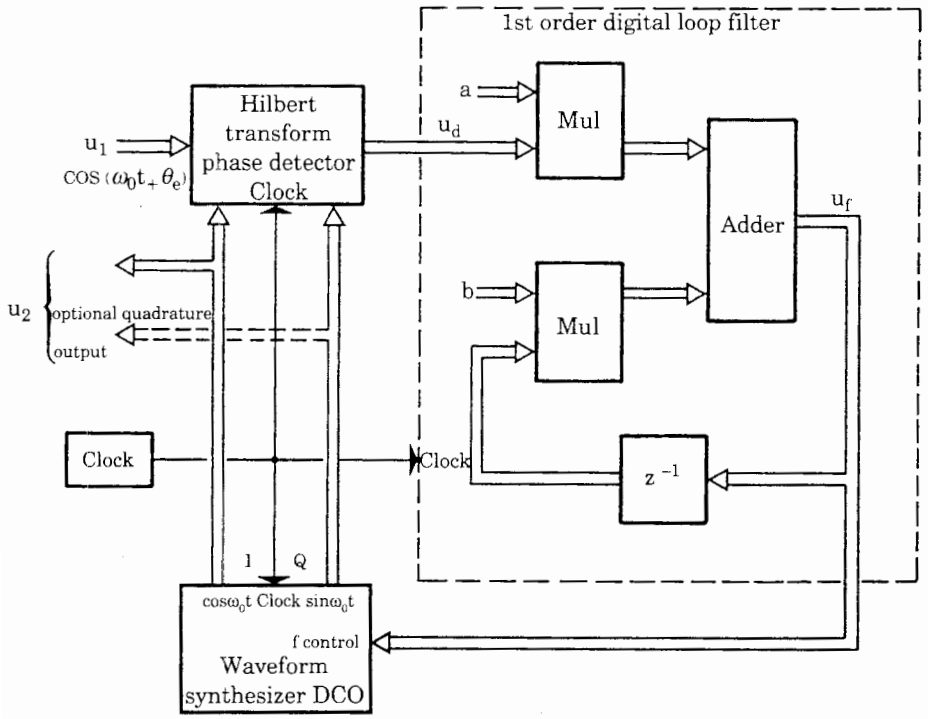


Figure 6.14 All-digital PLL system, example 3. This system is best implemented by software.

$$u_f(nT) = b_0 u_d(nT) - a_1 u_f[(n-1)T] \tag{6.10}$$

where a_1 and b_0 are filter coefficients as defined in Eq. (6.7). The mathematical operations performed by the digital filter are represented by the dashed block in Fig. 6.14.

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.[30] This would yield a third-order PLL.

6.3 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 like LPLLs. Others will operate like classical DPLLs, but the functioning of many ADPLLs will have almost nothing in common with LPLLs 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 (e.g., phase-transfer functions), Bode diagrams, or the like. In many cases, 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 74xx297, which was already shown in Fig. 6.13.

6.3.1 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 phase detector, as shown in Fig. 6.13. Performance of the ADPLL is most conveniently analyzed by the waveforms of the circuit, which are shown in Fig. 6.15. 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 chosen 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 Kup and Kdn, 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.

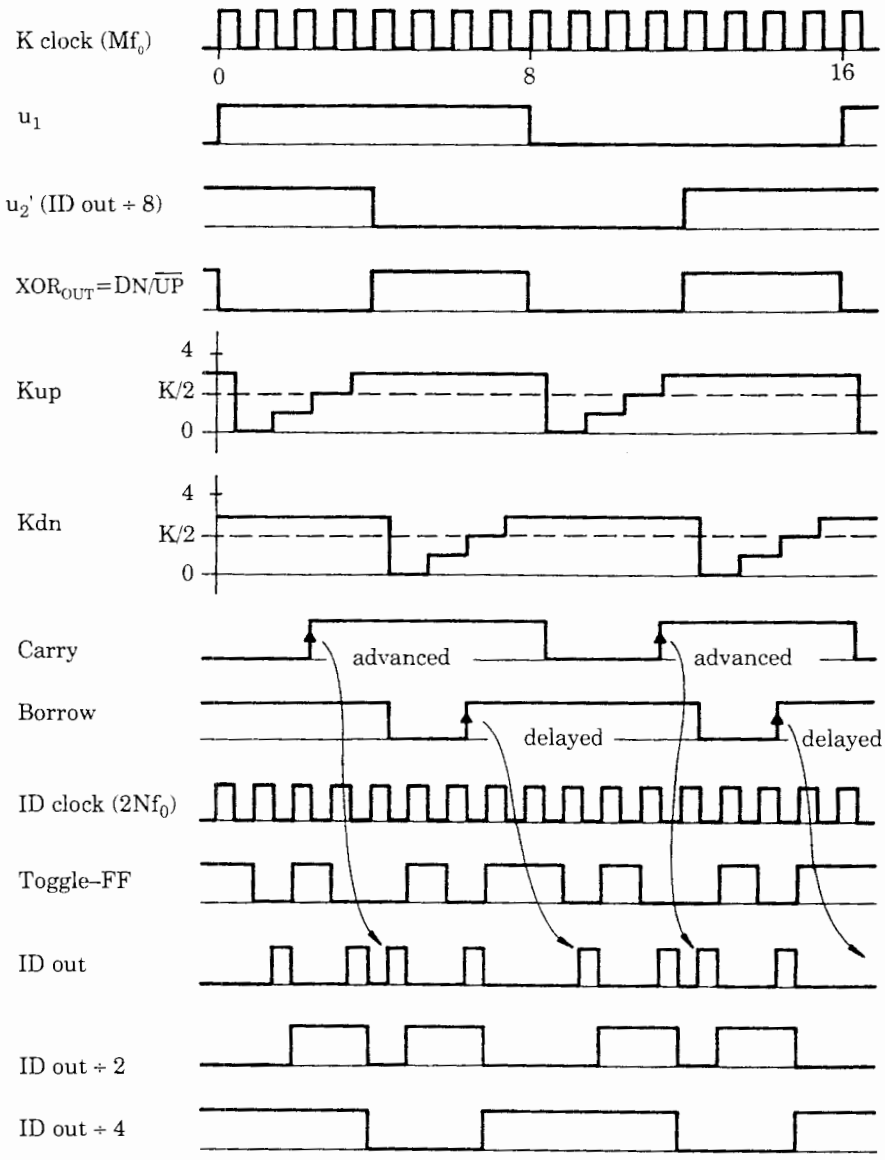


Figure 6.15 Waveforms of an ADPLL system using the IC type 74HC/HCT297. The ADPLL operates at its center frequency. The EXOR PD is used, and the parameters of the circuit are $M = 16$, $K = 4$, and $N = 8$. Note that K has been chosen small to simplify the drawing; in a real application, however, K cannot be less than 8.

At the instant where the first (0th) K clock occurs (Fig. 6.15), Kup and Kdn 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; i.e., the first stage of the $\div N$ counter (denoted *IDout* $\div$ *2* in Fig. 6.15) counts onto the negative edge of IDout, the second stage (denoted *IDout* $\div$ *4*) counts onto the negative edge of the IDout $\div$ 2 output signal, etc.

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. 6.15 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 IDout waveform shows, its pulses are periodically advanced and delayed by one cycle of the ID clock. The bottom-most two signals represent the scaled-down IDout 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 IDout signal does not have constant frequency but rather exhibits phase jitter. The output signals IDout $\div$ 2 and IDout $\div$ 4 are also asymmetrical, but the output signal u_2' (which corresponds to IDout $\div$ 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 IDout 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, e.g., $K = 8$. All other parameters remain unchanged. Figure 6.16 shows what happens now.

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 in one quarter-period of u_1 on average. Carries and borrows are now generated only in each second up-counting or down-counting interval. Figure 6.16 shows two periods of the reference signal, which correspond 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 7 K clock pulses instead of 8. In succeeding reference periods (not shown in this figure) there must be half-cycles of u_2' that have a duration of 9 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

$$K = M/4 \tag{6.11}$$

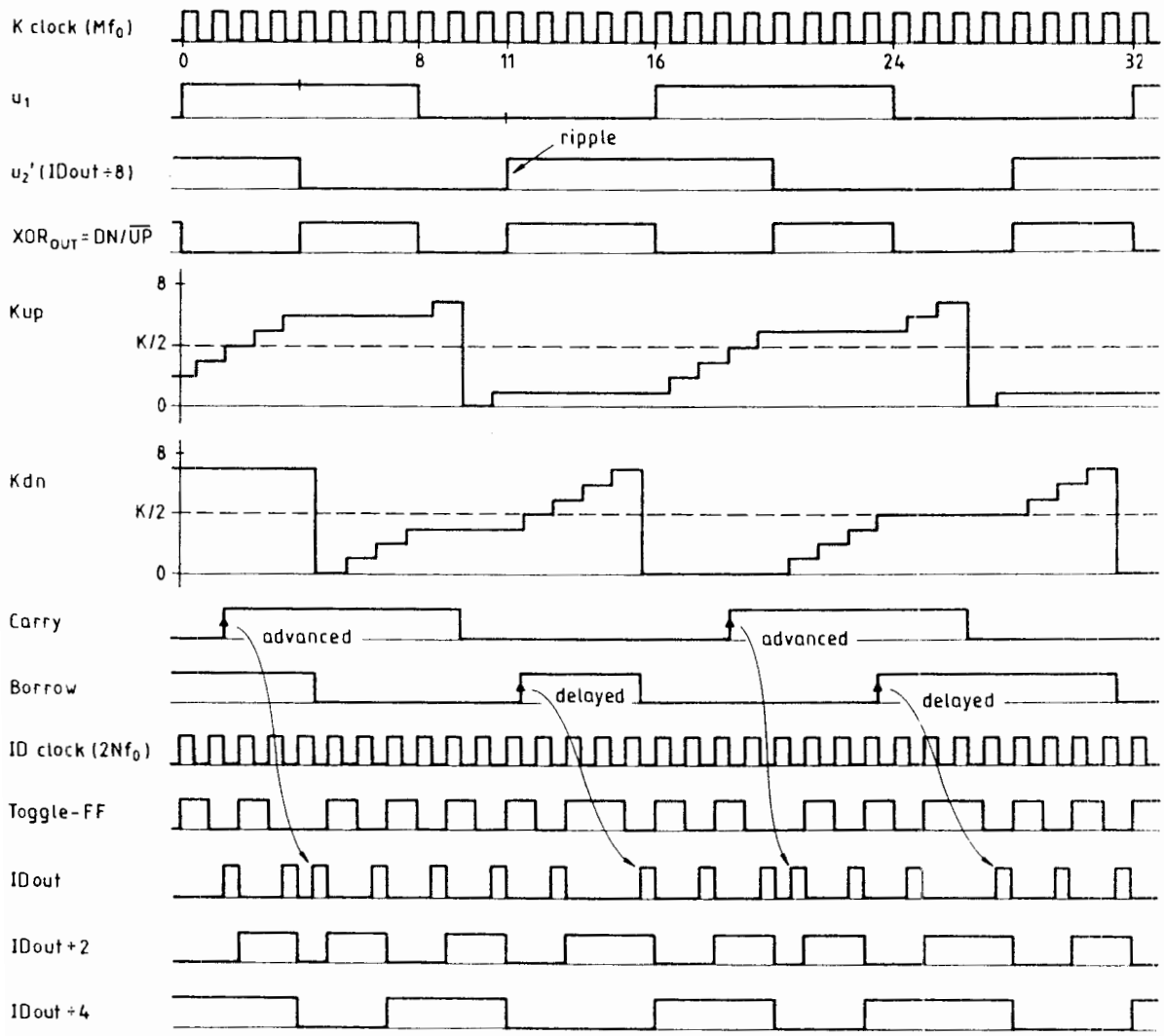


Figure 6.16 Conditions are like those is Fig. 6.15, but the parameters of the ADPLL are $M = 16$, $K = 8$, and $N = 8$. The ADPLL operates at its center frequency. Note that the up counter and down counter of the K counter recycle only on every second up-counting or down-counting period, respectively.

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 δ 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, i.e., by a time interval of $1/(2Nf_0)$. Consequently, the actual duty factor can vary in the range

$$0.5\left(1-\frac{1}{N}\right)<\delta<0.5\left(1+\frac{1}{N}\right) \tag{6.12}$$

i.e., 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 us check what happens if K is chosen smaller than $M/4$. If we consider Fig. 6.15 again and assume $K = 2$ now (which is not possible, however, with the 74HC/HCT297 IC), the up counter would generate 2 carries in one up-counting period, and the down counter would generate 2 borrows in one down-counting period. Because 2 carries and 2 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, however, only 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 3 ID clock periods, as shown in Sec. 6.1.3 (see also Fig. 6.10). Because the duration of one ID clock cycle is $1/(2Nf_0)$ seconds, we have no overslept carries (or borrows) when the condition

$$N > N_{\min} = \frac{3M}{2K} \tag{6.13}$$

is met. Now M, K, and N are mostly integer powers of 2, so in practical applications the minimum N will be chosen

$$N > N_{\text{pract}} = \frac{2M}{K} \tag{6.14}$$

where N_{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 ($K \le M/4$) and when N is chosen such that all carries and borrows can be processed ($N \ge 2M/K$). 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. (6.12).

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. 6.13, respectively, and that the Q output

of this flipflop is tied with the DN/$\overline{\text{UP}}$ input of the K counter. As we know from 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. 6.17 for the parameters $M = 16$, $K = 8$, and $N = 8$. If both signals u_1 and

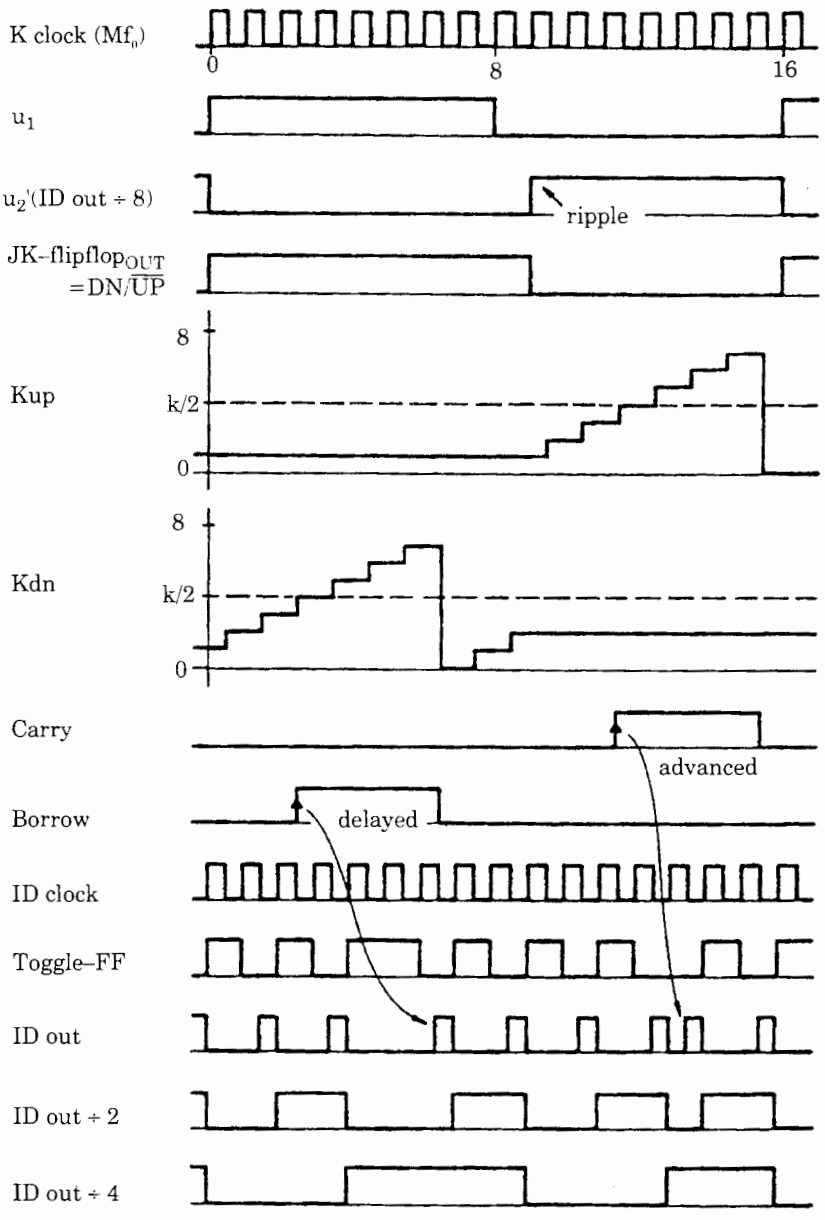


Figure 6.17 Conditions are like those in Fig. 6.15, but the JK-flipflop phase detector is used. The ADPLL operates at its center frequency. The parameters of the circuit are $M = 16$, $K = 8$, and $N = 8$.

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

$$0.5\left(1-\frac{M}{2KN}\right)<\delta<0.5\left(1+\frac{M}{2KN}\right) \tag{6.15}$$

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

$$K=\frac{M}{2} \tag{6.16}$$

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, $M = 2N$; i.e., the K clock and the ID clock are taken from the same generator. Then the duty cycle is in the range

$$0.5\left(1-\frac{1}{2K}\right)<\delta<0.5\left(1+\frac{1}{2K}\right) \tag{6.17}$$

Selecting a large value for K reduces the ripple accordingly.

6.3.2 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 phase detector. Furthermore, let the reference frequency be $f_1 = 1.25\ f_0$. Figure 6.18 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 become positive. As Fig. 6.18 shows, the signals u_1 and u_2' are nearly in phase now, which corresponds to a phase error near 90°. 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

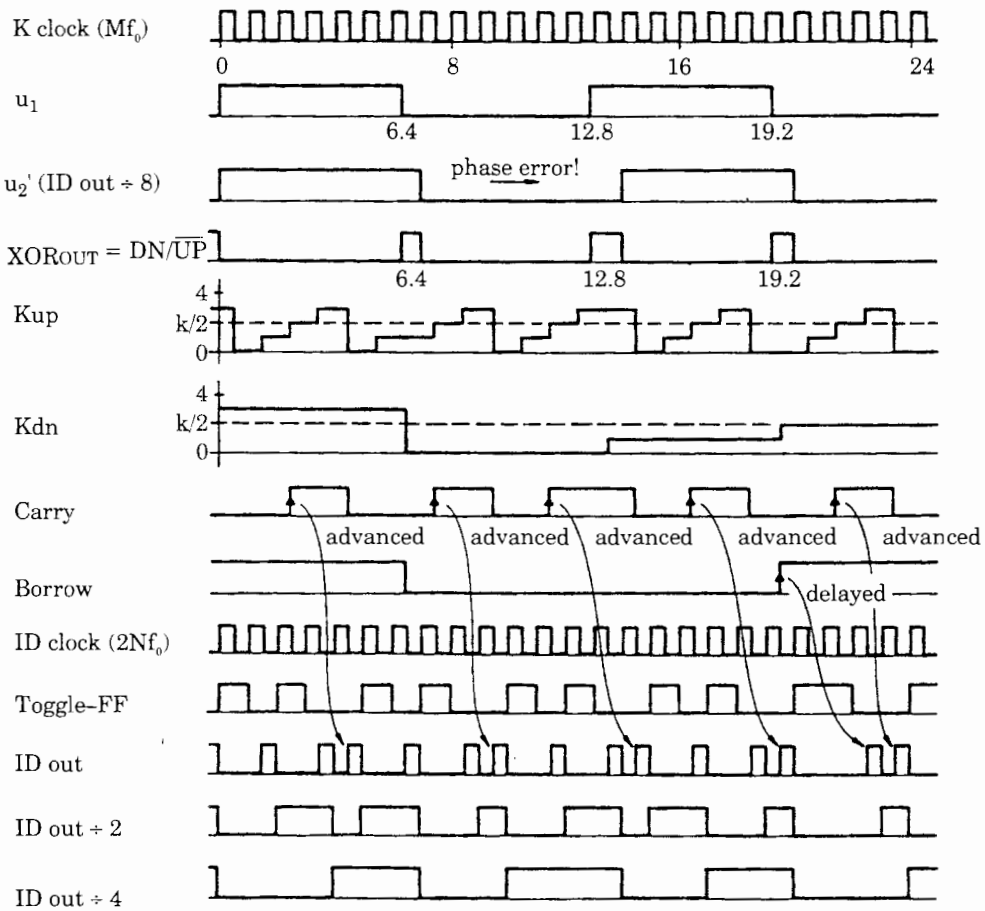


Figure 6.18 Conditions are like those in Fig. 6.15. The reference frequency is 1.25 times the center frequency here. Consequently, the up-counting section of the K counter is active most of the time, and more carries than borrows are generated on average.

when the K counter is continually counting up. The frequency of carry pulses then is given by

$$f_{\max} = f_0 \frac{M}{K} \tag{6.18}$$

Because each carry applied to the increment input of the ID counter causes $\frac{1}{2}$ cycle to be added to the IDout signal (refer to Sec. 6.1.3 and Fig. 6.10), the frequency at the output of the ID counter is increased by

$$\Delta f_{\text{IDout}} = f_0 \frac{M}{2K} \tag{6.19}$$

Because the $\div N$ counter scales down that frequency by N, the maximum frequency deviation from the center frequency the ADPLL can handle is

$$\Delta f_H = f_0 \frac{M}{2KN} \tag{6.20}$$

This is nothing else than the *hold range* of the ADPLL. The hold range given by Eq. (6.20) is realizable, however, only if N is chosen larger than $N_{\min}$ defined in Eq. (6.13). If N is smaller than $N_{\min}$, some of the carries and borrows are overslept, and the hold range is limited to

$$\Delta f_H = \frac{f_0}{3} \tag{6.21}$$

A simple consideration shows that the hold range given by Eq. (6.20) or (6.21) is a theoretical limit, which is not realized in practice. For the parameters M = 16, K = 4, and 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. 6.18. 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, . . . , 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 from Fig. 6.18. 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 LPLLs 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 that 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 in the order of the period of the reference signal, and hence is 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, pullout range, pull-in range, and hold range are all about the same for this circuit. We perform some ADPLL simulations in Chap. 7.

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 = integer, then we have $mT_0 = nT_1$, where $T_0 = 1/f_0$ and $T_1 = 1/f_1$; that is, 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.

6.3.3 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. 6.13) must be found. The model is shown in Fig. 6.19. The phase detector represents a zero-order block with gain K_d. The phase detector output signal controls the duty factor δ_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, δ_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

$$K_d = \frac{2}{\pi} \tag{6.22a}$$

For the JK-flipflop, the phase detector gain is

$$K_d = \frac{1}{\pi} \tag{6.22b}$$

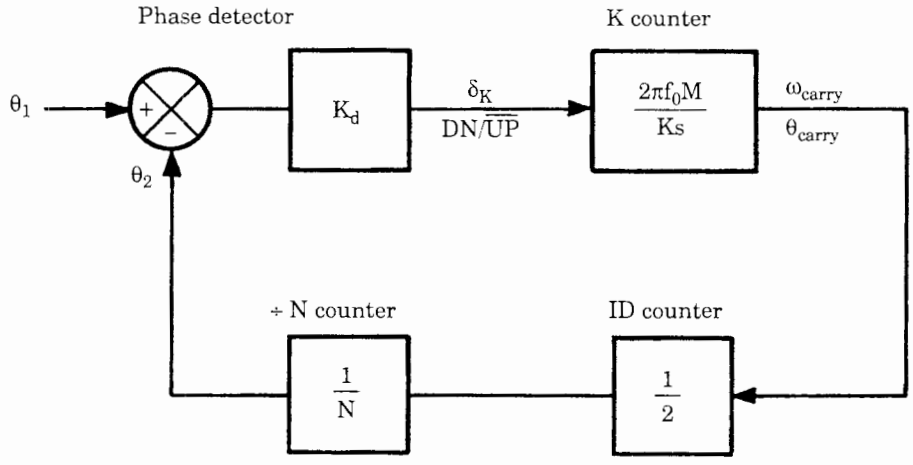


Figure 6.19 Mathematical model of the ADPLL. Definitions of symbols in text.

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. 6.3.2, the number of carry pulses generated per second is given by

$$f_{\text{carry}} = \delta_K \frac{Mf_0}{K} \tag{6.23}$$

The corresponding angular frequency ω_{carry} is therefore

$$\omega_{\text{carry}} = \delta_K 2\pi \frac{Mf_0}{K} \tag{6.24}$$

Because we are looking for the phase-transfer function, we must know the phase θ_{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

$$K_K(s) = \frac{\Theta_{carry}(s)}{\Delta_K(s)} = \frac{2\pi Mf_0}{Ks} \tag{6.25}$$

where $\Delta_K(s)$ and $\Theta_{\text{carry}}(s)$ are the Laplace transforms of the signals δ_K and θ_{carry}, respectively. Because each carry pulse applied to the incremental input of the ID counter causes $\frac{1}{2}$ cycle to be added to the IDout signal, the ID counter can be modeled simply by a zero-order block having the gain $\frac{1}{2}$. Clearly, the $\div N$ counter is a block with gain $1/N$. Having the model, we now find the phase-transfer function $H(s)$ to be

$$H(s) = \frac{\omega_0}{s + \omega_0} \tag{6.26}$$

where ω_0 is given by

$$\omega_0 = \frac{K_d \pi Mf_0}{KN} \tag{6.27}$$

Moreover, the error-transfer function $H_e(s)$ is

$$H_e(s) = \frac{\omega_0}{s + \omega_0} \tag{6.28}$$

Clearly, this ADPLL is a first-order system. Its time constant is given by

$$\tau = \frac{1}{\omega_0} = \frac{KN}{K_d \pi Mf_0} \tag{6.29}$$

Thus for the EXOR phase detector, the time constant of the ADPLL becomes

$$\tau(\text{EXOR}) = \frac{KN}{2Mf_0} \tag{6.30a}$$

and for the JK-flipflop phase detector

$$\tau(\text{JK}) = \frac{KN}{Mf_0} \tag{6.30b}$$

If the K modulus is chosen for minimum ripple, i.e., $K = M/4$ 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, its response onto phase or frequency steps becomes slower. Some numerical examples will be presented in Chap. 7.

6.3.4 Ripple reduction techniques

We saw in Sec. 6.3.1 that ripple in ADPLLs can be minimized by choosing an optimum value for the K counter modulus; i.e., $K = M/4$ when the EXOR PD is used or $K = M/2$ when the JK-flipflop PD is used. There are other simple ways to reduce ripple. Figure 6.20*a* shows a ripple cancellation scheme which uses a feature of the K counter that has not yet been mentioned: the enable (EN) input of the K counter. Only one additional inverter is required in this circuit. The K counter is in operation only when 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/$\overline{\text{UP}}$ input of the K counter, but rather to its enable input EN. The DN/$\overline{\text{UP}}$ 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. 6.20*b*). 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' established. Figure 6.20*c* shows the case for maximum phase error θ_e. Now, the EN signal is high 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. 6.13). Consequently, the hold range is reduced roughly by a factor of 2. It has been shown[16] that the hold range of the circuit becomes

$$\Delta f_H = \frac{Mf_0}{2N(2K+1)} \tag{6.31}$$

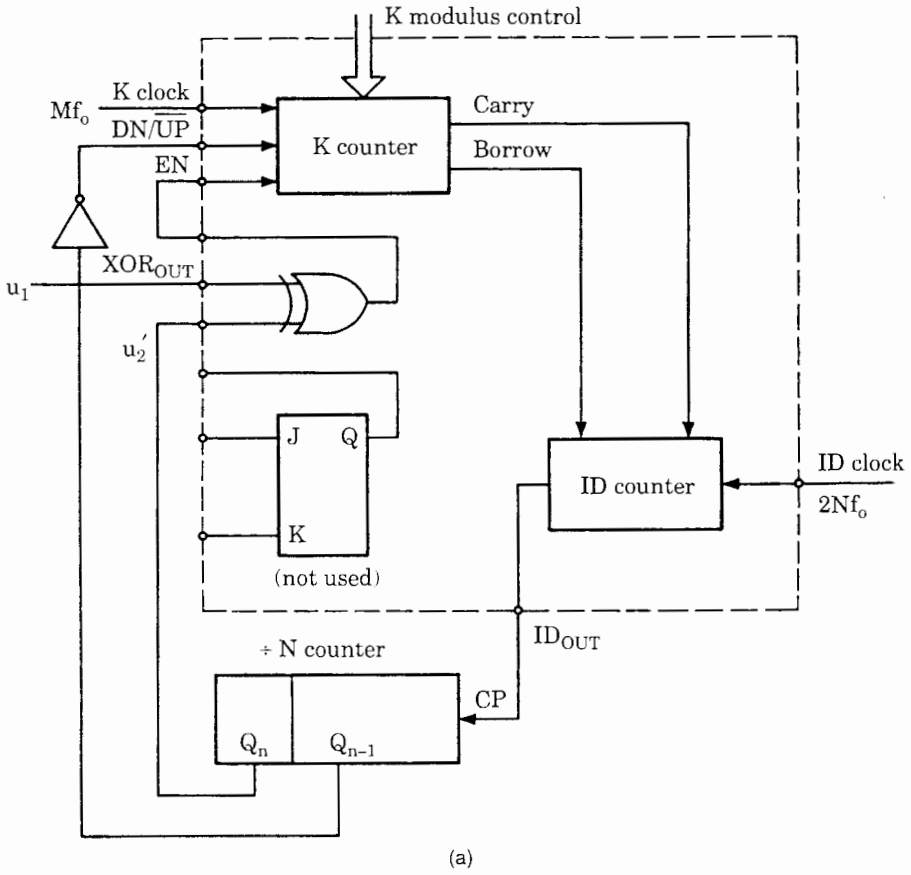


Figure 6.20 Ripple cancellation scheme for an ADPLL. (*a*) This circuit makes use of the enable (EN) input of the K counter. For details, refer to text. (*b*) The ADPLL operates on its center frequency. (*c*) The reference frequency is higher than the center frequency.

If $M = 2N$ and K is large, this reduces to

$$\Delta f_H \approx \frac{f_0}{2K} \tag{6.32}$$

Still other ripple cancellation schemes are discussed in Ref. 16.

6.3.5 Higher-order ADPLLs

The ADPLL considered in this section is a first-order system. As we have seen in Eqs. (6.30), its settling time can be made extremely short. This can be an advantage in some applications; there are other cases, however, where a slower 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[16] in some more detail.

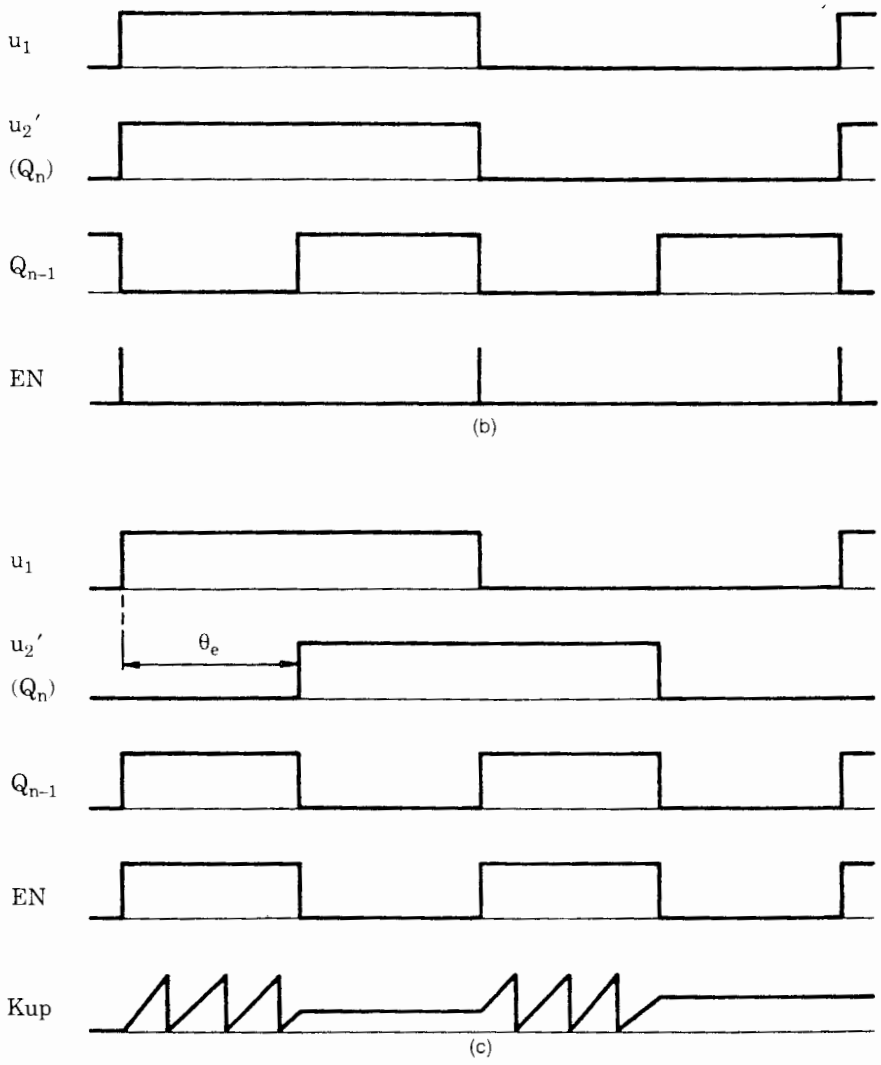


Figure 6.20 *(Continued)*

6.4 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 will be considered in the following. FSK is the abbreviation for frequency shift keying. In FSK data transmission, serial binary data are transmitted by using two different frequencies, one of which represents a logical 0, the other a logical 1.

An extremely simple FSK decoder circuit is shown in Fig. 6.21. 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

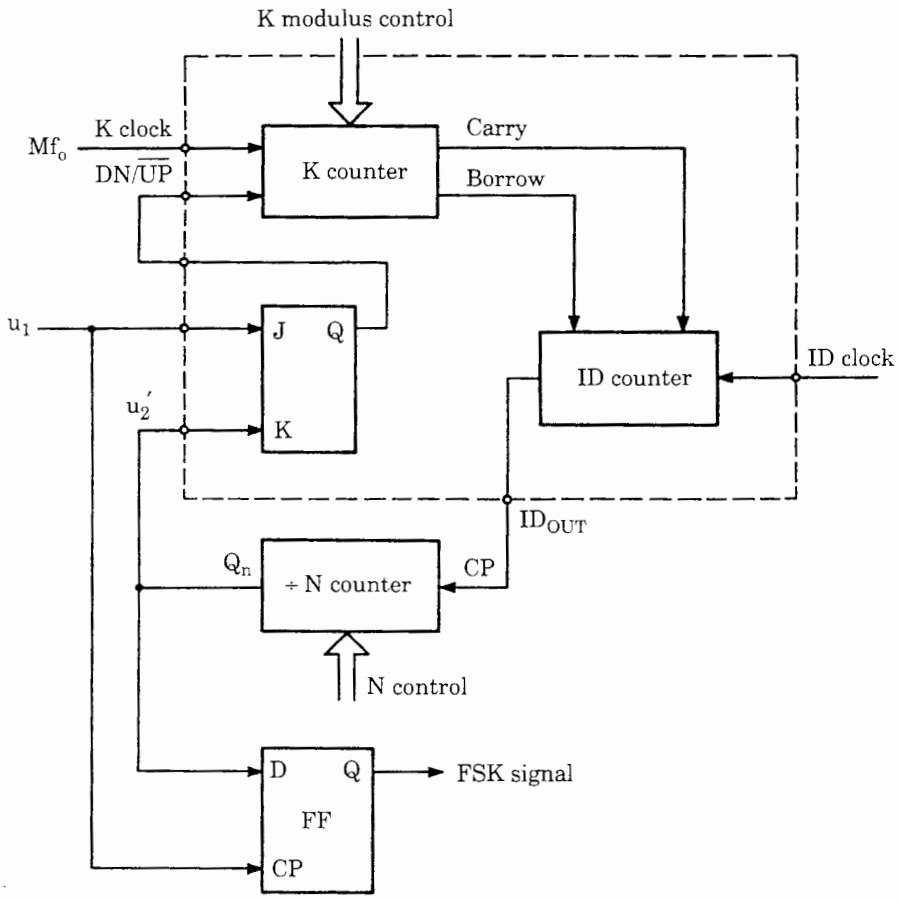


Figure 6.21 FSK decoder using the 74HC/HCT297 ADPLL IC. An additional D-flipflop is used to deliver the demodulated FSK signal.

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. 6.17). 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 content 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 D-flipflop every time the reference signal goes high. This D-flipflop is shown at the bottom of Fig. 6.21. In Sec. 6.5 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[31] and in various application notes delivered by the IC suppliers.[9,16]

6.5 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. 6.13, 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 be used only 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. 6.3, there are a number of interdependencies 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 "oversleeping" of carries and borrows, N should be greater than a minimum value N_{min}, which is given by Eq. (6.13). As indicated by Eq. (6.20*a*), the resulting hold range is a function of M, K, and N. Furthermore, the settling time τ of the ADPLL also depends on these parameters, as seen from Eqs. (6.30*a* and *b*).

On the basis of these general considerations, we can derive the design procedure given in Fig. 6.22. 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. 6.4; refer also to Fig. 6.21.

6.5.1 Case study: Designing an ADPLL FSK decoder

Step 1. To specify the parameters of this ADPLL system, we must define first its center frequency and hold range. Let us assume that the FSK transmitter uses the frequencies $f_{11} = 2100\,\text{Hz}$ and $f_{12} = 2700\,\text{Hz}$ to encode the binary information 0 and 1, respectively. Following the description in Sec. 6.4, we choose a center frequency of $f_0 = 2400\,\text{Hz}$. To ensure that both frequencies of the FSK transmitter are within the hold range of the ADPLL, we specify a hold range of $\Delta f_H = 600\,\text{Hz}$. We do not specify a settling time τ for the moment but will check at the end of the design whether the resulting settling time is appropriate or not.

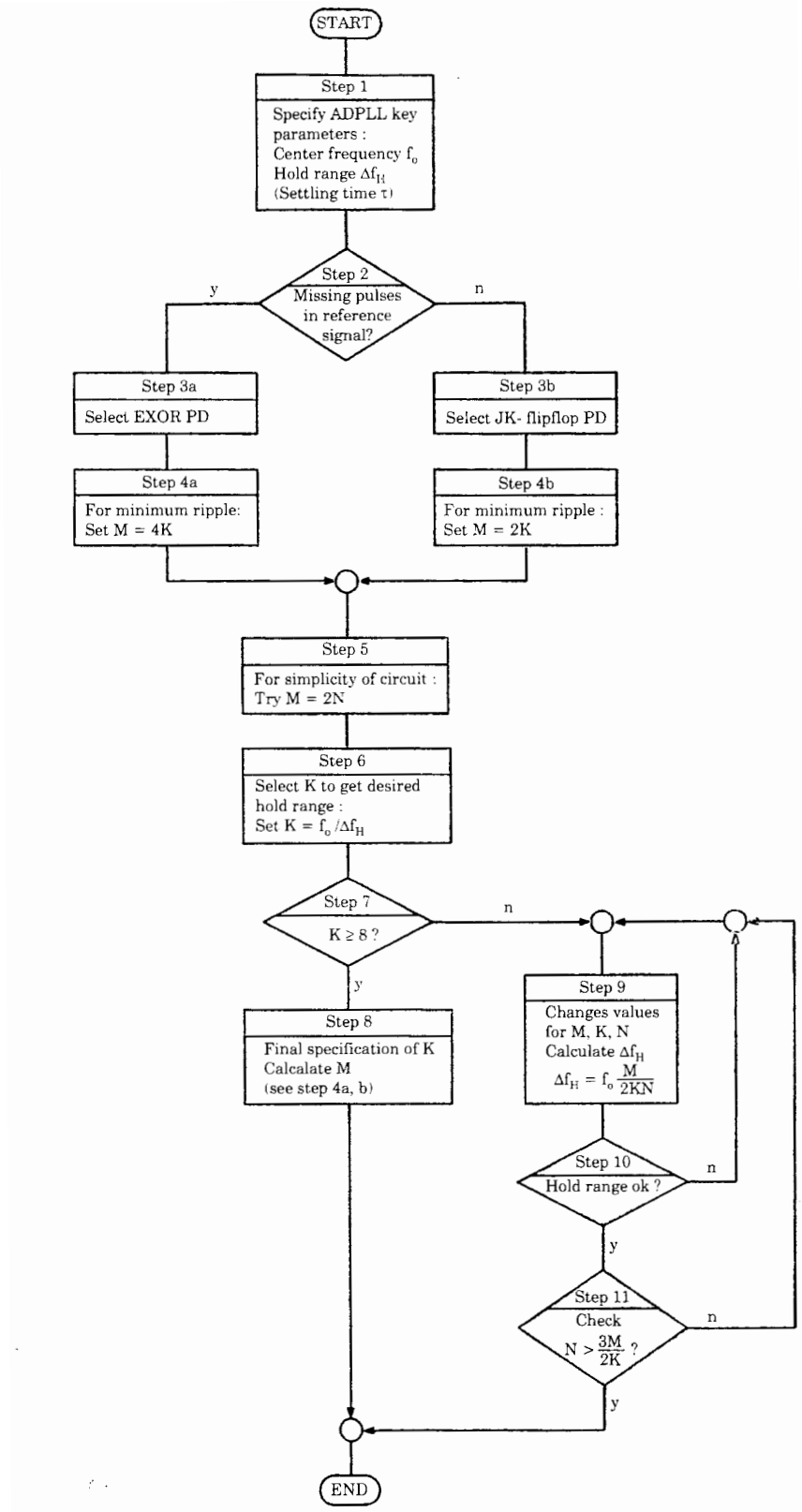


Figure 6.22 Design procedure for the ADPLL.

Step 2. The decision of whether the EXOR or the JK-flipflop will be used as phase detector has been made in advance. Let us keep in mind, however, that this ADPLL relies on an input signal which does not show up with 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 $K = 8, 16, \ldots$. 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 will keep $M = 2K$ (the condition for minimum ripple), the hold range can be expressed as

$$\Delta f_H = f_0 \frac{M}{2KN} = f_0 \frac{1}{N} \tag{6.33}$$

The hold range 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 which scales down the K clock frequency by a factor of 2.

Step 10. The hold range is OK 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 τ of this circuit. Using Eq. (6.30*b*), 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 (see Chap. 7), we look at this circuit and see how it performs.

Chapter

7

Computer-Aided Design and Simulation of ADPLLs

7.1 Setting up the Design Parameters

The program distributed with the book can also be used to design and simulate ADPLLs. The procedures are similar to those described in Chap. 5 and are best explained by a quick tour.

Start the program and hit the *CONFIG* speedbutton in the taskbar. This brings up the configuration dialog box, Fig. 7.1. In the panel *PLL Category*, click the *ADPLL* radiobutton. To configure an ADPLL, it is required only to specify the center frequency *f0* and the type of phase detector, *Exor* or *JK-FF*. Hit the *Done* button to complete the dialog.

Next click the *DESIGN* speedbutton in the taskbar. This activates the design dialog box, Fig. 7.2. In the top-most panel (*Configuration*) the actual ADPLL configuration is displayed. Below there are two panels, labeled *Frequency params* and *ADPLL params*. The ADPLL params are represented by the parameters K, M, and N as described in Sec. 6.3. For the last-specified values for K, M, and N, the program computes the corresponding ADPLL hold range Δf_H and 3-dB corner frequency f_{3dB}. As in case of the mixed-signal PLL, the user can proceed in two ways: (1) Enter the desired hold range into the edit window labeled *fH* and then hit the *Calc ADPLL params* button. The program then computes the parameters K, M, and N required to meet that goal. (2) Alternatively, immediately enter values for K, M, and N into the corresponding edit windows and hit 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. (6.20). Because we have only one equation for three unknown parameters, two of them can be chosen arbitrarily. The program therefore sets $K = 8$ and $N = 16$ and finally computes the appropriate value of M. This is not necessarily the 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

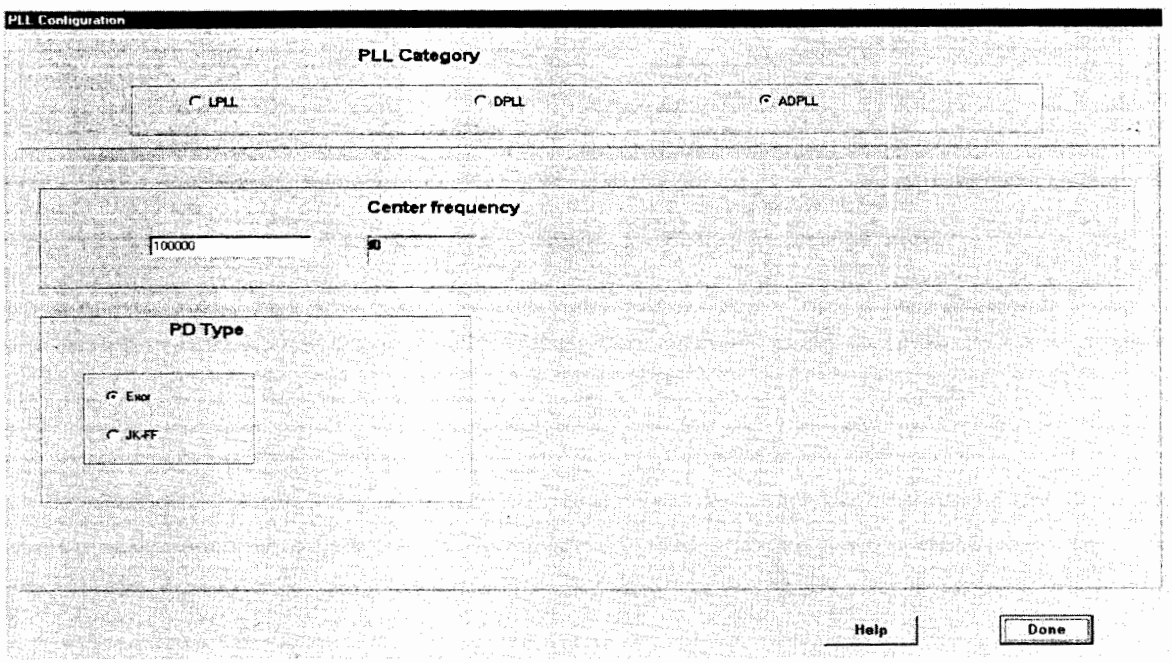


Figure. 7.1 Dialog box for configuring an ADPLL.

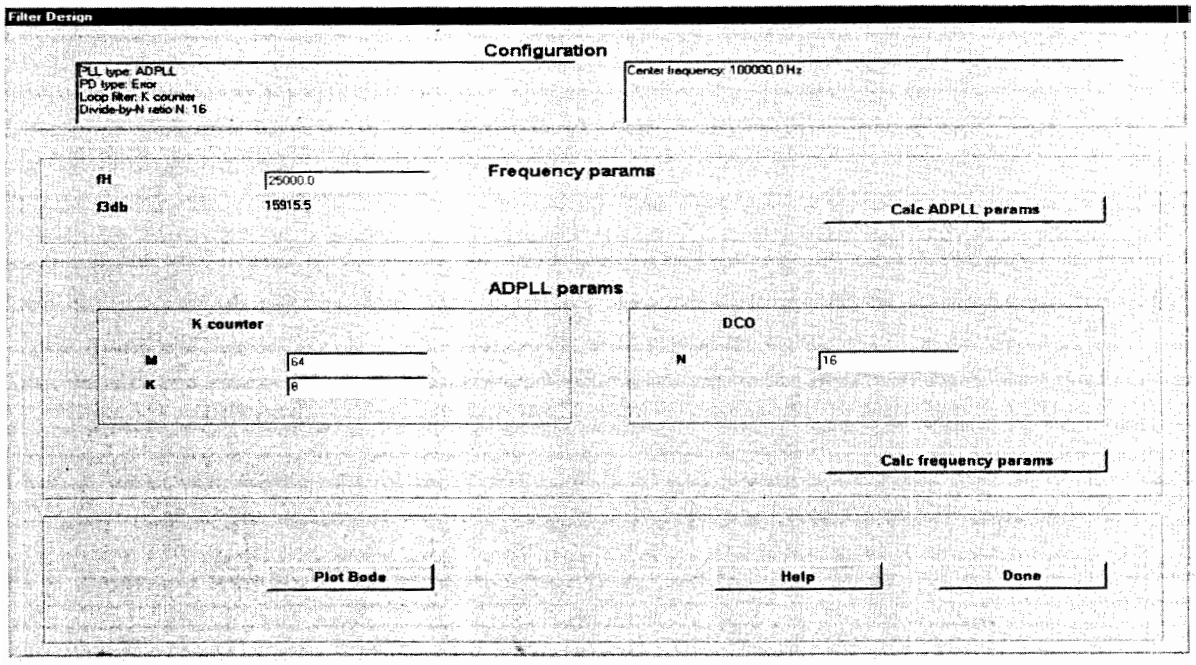


Figure. 7.2 Dialog box for ADPLL parameter specification.

$N = 32$. Because the hold range is proportional to M/KN, M would have to be increased by the same factor, i.e., 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 *Plot Bode* button. The *DESIGN* dialog exits when the *Done* button is hit. We are ready now to perform simulations.

7.2 Simulating ADPLL Performance

To start ADPLL simulation, hit the *SIM* speedbutton in the taskbar. This opens the simulation dialog box, Fig. 7.3. The parameters of the simulation are entered in the panel on the right side. They are almost identical with those of LPLL and DPLL 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' (see Fig. 6.13) and the instantaneous phase error θ_e. The frame window on the left of the simulation dialog box displays θ_e (thicker curve) and Δf_2 (thinner curve), where Δf_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. Because the output signal u_2' can change state only on the edges of the clock signal (ID clock in Fig. 6.13), only a discrete set of output frequencies can be created. As we see from the plot in Fig. 7.3,

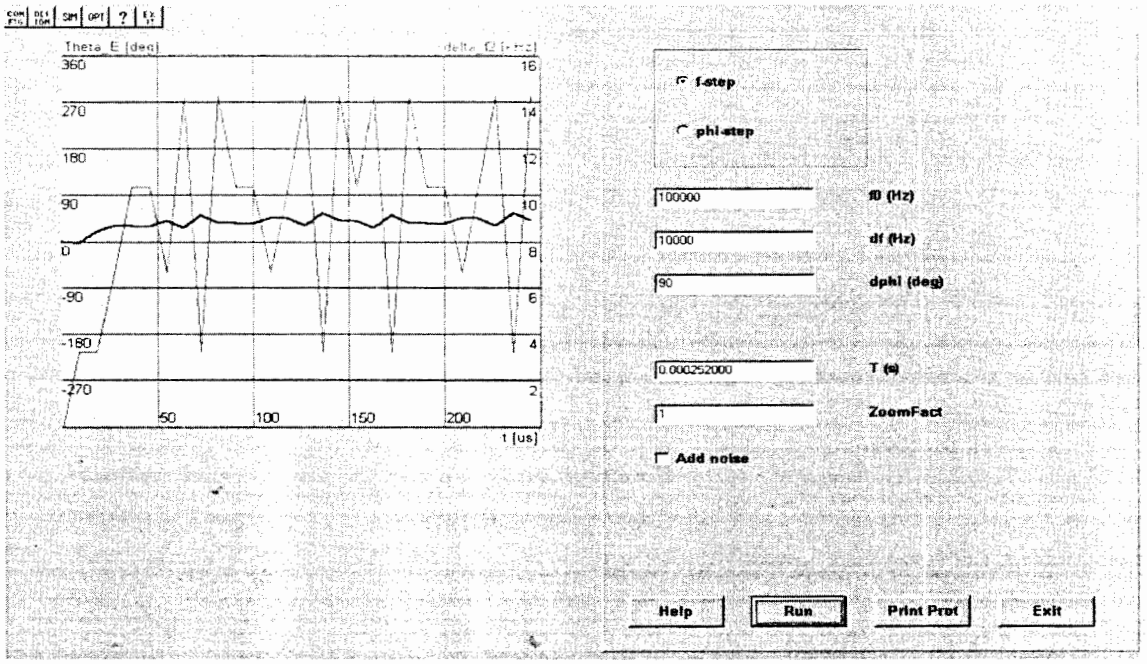


Figure. 7.3 Simulation of ADPLL performance.

the output frequency varies in steps of approximately 3.4 kHz. Because one single value for θ_e and Δf_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. 5.3 (see also Fig. 5.14).

In the next section we will perform a number of case studies that will reveal the substantial differences in the behavior of ADPLLs compared with the classical LPLLs and DPLLs.

7.3 Case Studies of ADPLL Behavior

In this section we will investigate ADPLL performance in 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* radiobutton. In the *Center frequency* panel, enter 100,000 into the *f0* window. In the *PD Type* panel, check the *Exor* radiobutton. Close the *CONFIG* dialog by hitting the *Done* button.

Click the *DESIGN* speedbutton. This opens the design dialog box. In the *ADPLL params* panel, enter the values

$$K = 8$$
$$M = 32$$
$$N = 16$$

Hit the *Calc frequency params* button. The program computes the resulting hold range and 3-dB corner frequency. We get fH = 12.5 kHz and f_{3dB} = 7957 Hz. Hit the *Done* button to close the design dialog.

You are ready now for simulation. Click the *SIM* speedbutton. Because we want to apply frequency steps to the reference inputs, check the *f-step* radiobutton (see Fig. 7.3). In the edit window enter the following parameters

$f0 = 100{,}000$ (Hz)
$df = 6000$ (Hz)
$T = 0.00040$ (s)
$ZoomFact = 1$
Add noise: unchecked

Hit the *RUN* button. According to the theory of this type of ADPLL, the time constant of the loop is $\tau = 2T_0 = 20\,\mu s$, 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. 7.4.

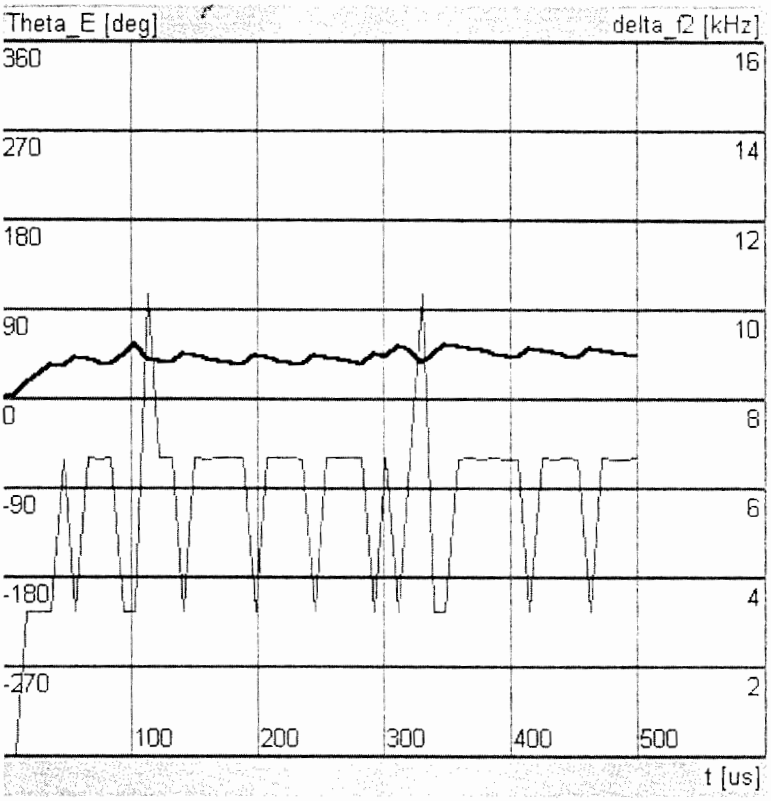


Figure. 7.4 ADPLL simulation. Response to frequency step df = 6 kHz. Thicker line = phase error; thinner line = frequency deviation.

A linear system would settle to within 5 percent of its final state in about three time constants, i.e., in 60 μs. The curve for the phase error is quite jittery, but the settling time is seen to be well within 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 perform state changes only 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

$$f_2 = f_0\left(\frac{2N}{2N \pm 1}, \frac{2N}{2N \pm 2}, \cdots\right) \tag{7.1}$$

For positive deviations, f_2 can have the values 100, 103.2, 106.7, . . . kHz. The curve for Δf_2 indicates that the output frequency equals the reference frequency *on average*, however. Because the phase-error curve is relatively smooth at about 45°, the ADPLL is firmly locked.

Let us increase now the frequency step to, say, 12 kHz. This simulation is shown in Fig. 7.5. 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, i.e., slightly less than 90° on average. If the frequency step is made larger (13 kHz), the system can no longer maintain lock, Fig. 7.6. The loop pulls the output frequency to about 114 kHz after some reference cycles, but because the phase error exceeds 180°, 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 us look now at how the ADPLL reacts to phase steps. Check the *phi-Step* radiobutton in the *SIM* dialog and enter *dphi* = 90°. Hit 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, Fig. 7.7.

Case study 2: *Dynamic performance of the ADPLL using the JK-flipflop PD.* Now we start another simulation and hit the *CONFIG* speedbutton first. In the *CONFIG* dialog box we check the *JK-FF* radiobutton. Enter a *Center frequency* of *f0* = 100 kHz

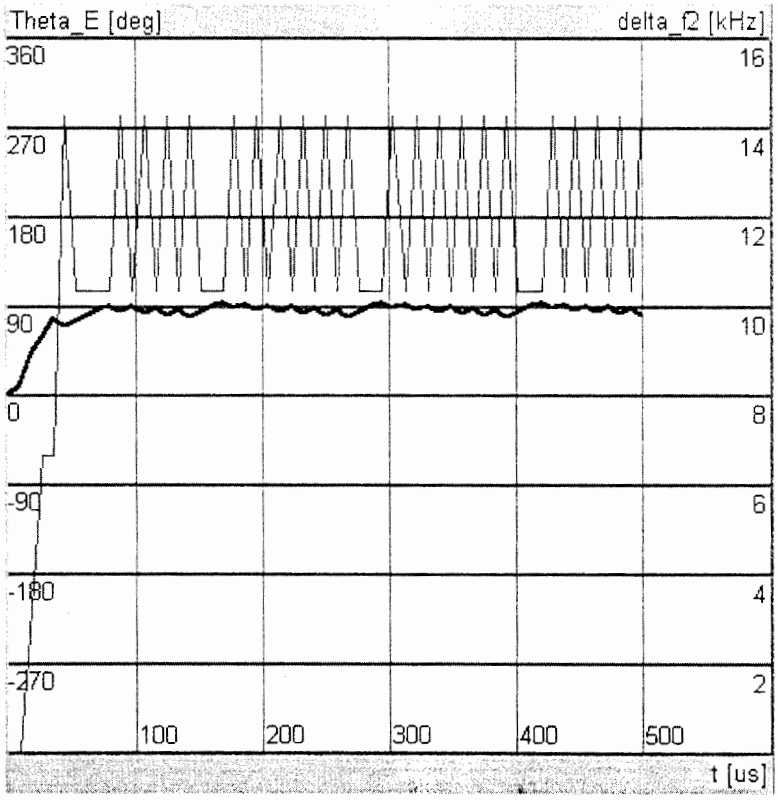


Figure. 7.5 ADPLL simulation. Response to frequency step *df* = 12 kHz.

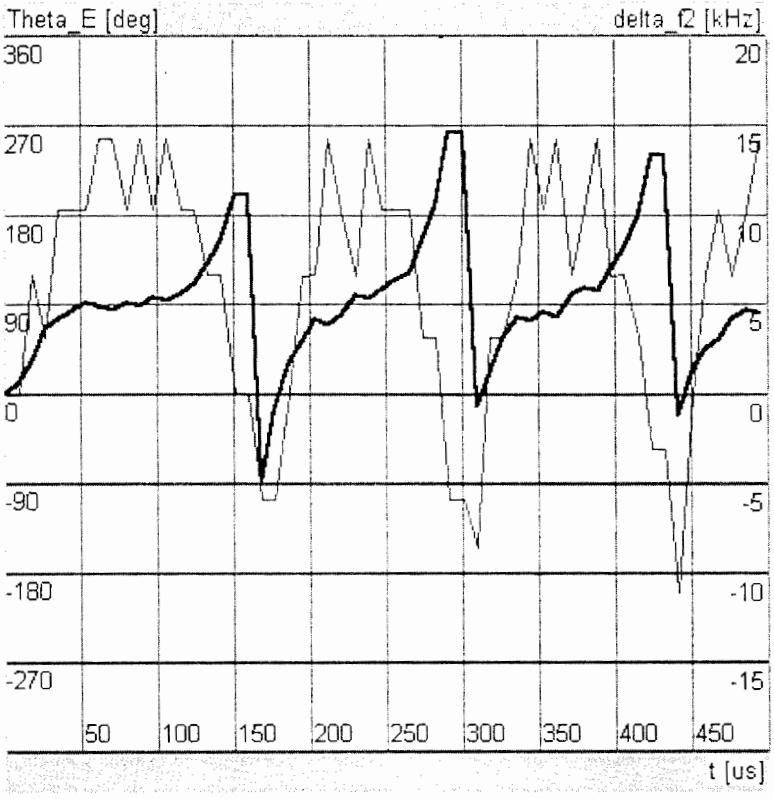


Figure. 7.6 ADPLL simulation. Response to frequency step $df = 13\,\text{kHz}$.

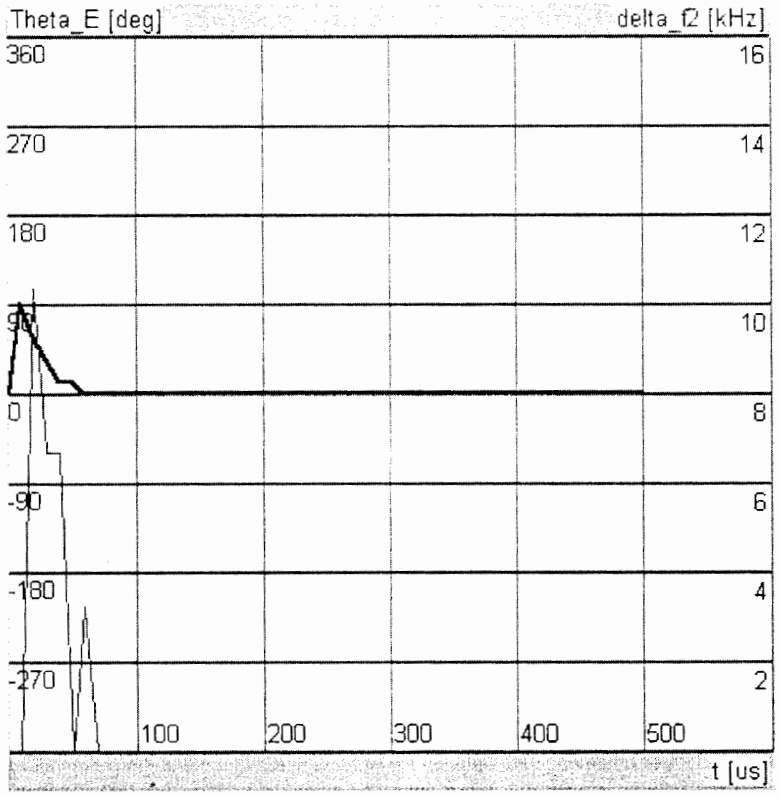


Fig. 7.7 ADPLL simulation. Response to phase step $\Delta\Phi = 90°$.

again. Hit the *Done* button to finish the *CONFIG* dialog and click the *DESIGN* speedbutton. In the *ADPLL params* panel, enter the parameters for minimum ripple, that is, $M = 16$, $K = 8$, and $N = 8$, and hit the *Calc frequency params* button. The program predicts a hold range of $fH = 12.5$ kHz. According to ADPLL theory, the time constant of the loop becomes $\tau = 40\,\mu s$. Hit the *Done* button to exit the *DESIGN* dialog and click the *SIM* speed button. In the *SIM* dialog, enter a frequency step $df = 11{,}000$ (kHz) and click the *RUN* button. The results of the simulation are shown in Fig. 7.8.

The loop locks, but the phase error nearly approaches the stability limit of 180°. 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 μs, which is roughly 3τ.

Case study 3: *Dynamic performance of the FSK decoder.* Let us investigate now the behavior of the FSK decoder we designed in Sec. 6.5. Start the simulation by clicking the *CONFIG* speed button. In the *PD type* panel, check the *JK-flipflop* radiobutton. In the *Center frequency* panel, enter $f_0 = 2400$ Hz. Hit the *Done* button to close the *CONFIG* dialog. Hit the *DESIGN* speed button to start the

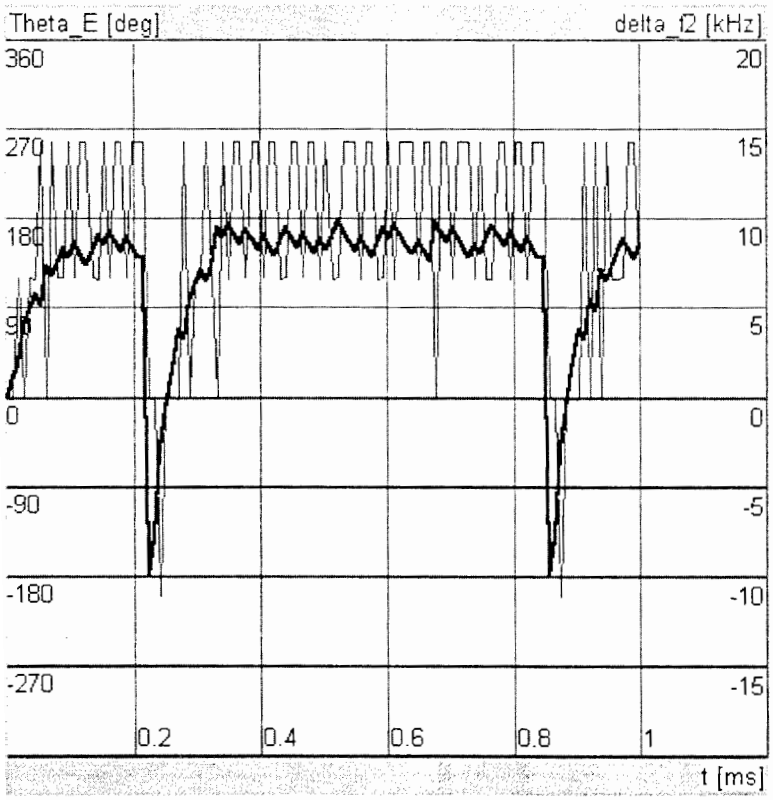


Figure. 7.8 ADPLL simulation. Response to frequency step $df = 11$ kHz. Phase detector = JK-flipflop.

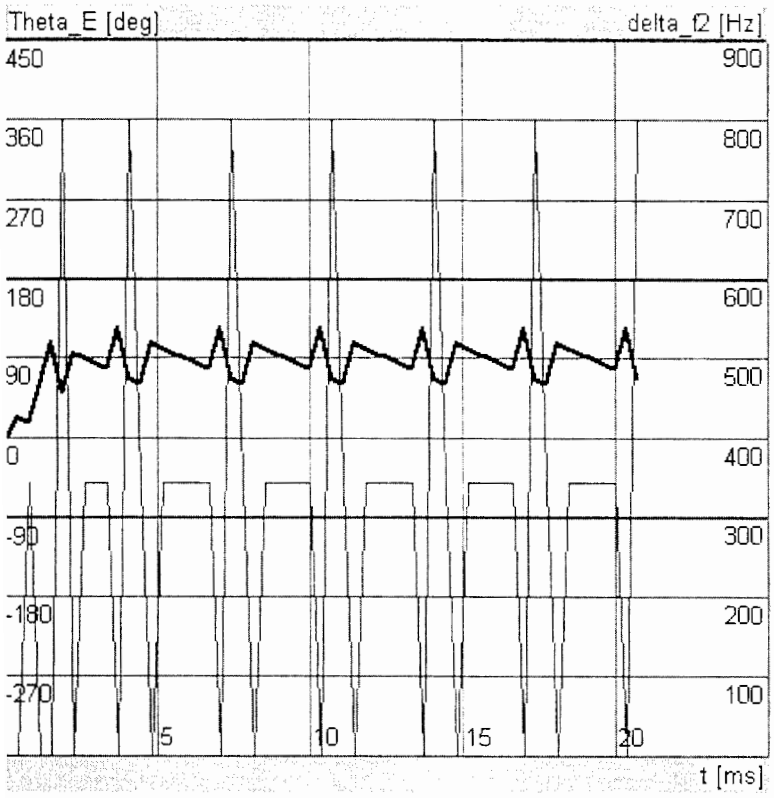


Figure. 7.9 Response of the FSK decoder onto a frequency step $df =$ 300 Hz.

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 hitting the *SIM* speed button. In the *SIM* dialog, check the *f-Step* radiobutton and specify a frequency step $df = 300$ Hz. Hit the *RUN* button to start the simulation. The results are shown in Fig. 7.9. Clearly, the ADPLL locks within a few reference cycles. The phase error is around 90^0. There is quite a bit of ripple on the output frequency, which is not critical in this application. As Eq. (6.15) indicates, the frequency ripple could be reduced by increasing N. Doubling N, for example, would reduce the hold range, however, 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 his or her own trials on improving the frequency ripple of this ADPLL ability to maintain lock.

Chapter

8

The Software PLL (SPLL)

8.1 The Hardware-Software Tradeoff

In the age of microcontrollers and digital signal processors (DSPs) it is an obvious idea to implement a PLL system by software. When that is done, the functions of the PLL are performed by a computer program. The designer realizing a software PLL (SPLL) 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 that is used to run the program. If a given algorithm performs too slowly on a relatively cheap microcontroller (one of the popular 8051 family, for example) and the designer is forced to resort to more powerful hardware (e.g., 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 by 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 µs for one pass-through.

Today's microcontrollers easily work with clock frequencies up to the gigahertz 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.[18] 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 instruction 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.

8.2 Feasibility of an SPLL Design

An SPLL design offers the most degrees of freedom available in any one PLL design, because it can be tailored to perform similarly to an LPLL or a DPLL or to execute a function that neither of these hardware variants is able to do. To check whether a software implementation can economically be justified, we recommend going through the steps described in the following.

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. 8.3. For the moment it is sufficient to write down these algorithms in symbolic form, i.e., 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. 8.3.

Step 2—Definition of the language. Having defined the algorithms, we should define the language that will be used to encode them, at least tentatively. The programming effort is minimized when a high-level language such as C/C++ 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,[18,19] the compiler must generate parallel assembly code, i.e., 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, e.g., 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 μs, for example, and two passes are required in one cycle of the reference signal, at least 100 μs of computation time is needed in one reference period. Probably the microcontroller or whatever hardware is used will need some more time for timekeeping, 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 that the real system is not even slower than the designer imagines.

8.3 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 adapt the methods to other SPLL realizations. The PLL simulation program on 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. We 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 many of the algorithms described in the following sections.

8.3.1 An LPLL-like SPLL

We are going to develop an SPLL algorithm that performs similarly to a hardware LPLL. To derive the required SPLL algorithm, we plot a signal flow diagram that shows the arithmetic operations within the loop (Fig. 8.1). The input signal u_1 is supposed to be an arbitrary analog signal, e.g., 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 $t = 0, T, 2T, \ldots, nT$. $u_1(n)$ is the simplified notation for the input signal sampled at time $t = nT$; i.e., $u_1(n) = u_1(nT)$. All other signals of the SPLL are sampled signals, too, and must be calculated at the sampling instants $t = 0, T, 2T, \ldots, nT$. Consequently, all function blocks of the signal flow diagram are working synchronously with the ADC clock.

In Fig. 8.1 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; i.e., $N = 1$.) The multiplier is used as phase detector and corresponds exactly with the already known Nyquist rate PD discussed in Sec. 6.1.1; refer also to Fig. 6.2. Its output signal is denoted $u_d(n)$. The digital filter serves as loop filter, 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 is denoted $u_2(n)$. As we will see, the

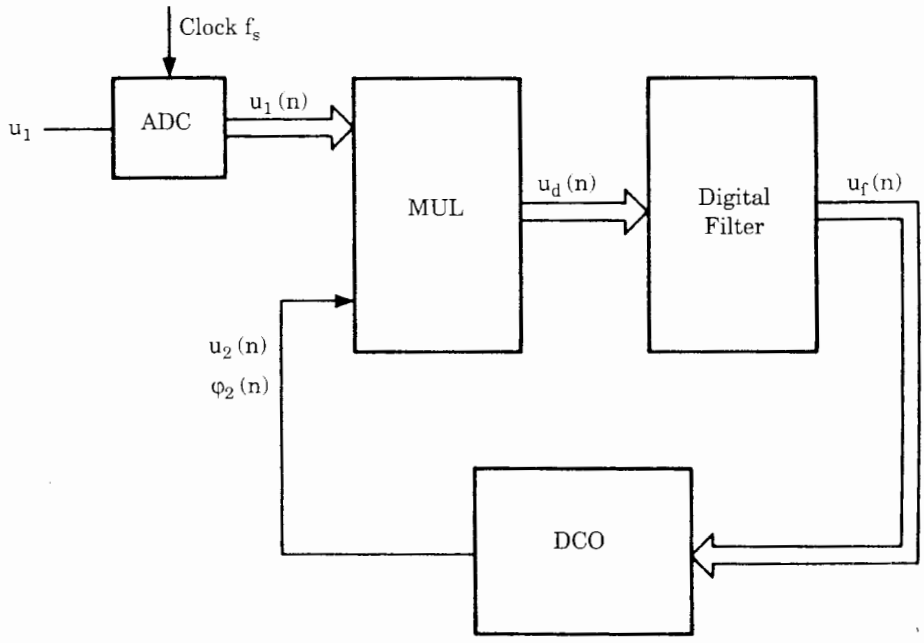


Figure 8.1 Block diagram showing the arithmetic operations to be performed by an SPLL whose performance is similar to the LPLL.

DCO is not able to compute $u_2(t)$ directly; rather, this signal must 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

$$\omega_2(t) = \omega_0 + K_0 u_f(t) \tag{8.1}$$

The continuous output signal $u_2(t)$ then would be given by

$$u_2(t) = \text{rect}(\omega_2(t) \cdot t) \tag{8.2}$$

where rect denotes the square wave function; refer to Eq. (2.10*b*). The total phase $\varphi_2(t)$ of the VCO output signal then would be

$$\varphi_2(t) = \int \omega_2(t)\, dt = \omega_0 t + K_0 \int u_f(t)\, dt \tag{8.3}$$

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. (8.3). 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 $2\pi < \varphi_2 < 3\pi$, etc. In all other cases, $u_2(t) = -1$. We are going now to 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

$$\Delta\varphi_2 = [\omega_0 + K_0 u_f(n)]T \tag{8.4a}$$

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

$$\varphi_2(n+1) = \varphi_2(n) + [\omega_0 + K_0 u_f(n)]T \tag{8.4b}$$

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 $t = 1 \cdot T$, etc. Given $\varphi_2\ (n + 1)$, we can also extrapolate the value of $u_2\ (n + 1)$ at $t = (n + 1)T$,

$$u_2(n+1) = 1 \quad \text{if} \quad 2k\pi \le \varphi_2(n+1) < (2k+1)\pi$$

or

$$u_2(n+1) = -1 \quad \text{if} \quad (2k-1)\pi \le \varphi_2(n+1) < 2k\pi$$

with k = integer.

The signals of our SPLL are depicted in Fig. 8.2. The dashed lines represent continuous signals. The sampled signals are plotted as dots. Only the continu-

ous signal $u_1(t)$ really exists; all others are only 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

$$u_d(n) = K_d u_1(n) u_2(n)$$

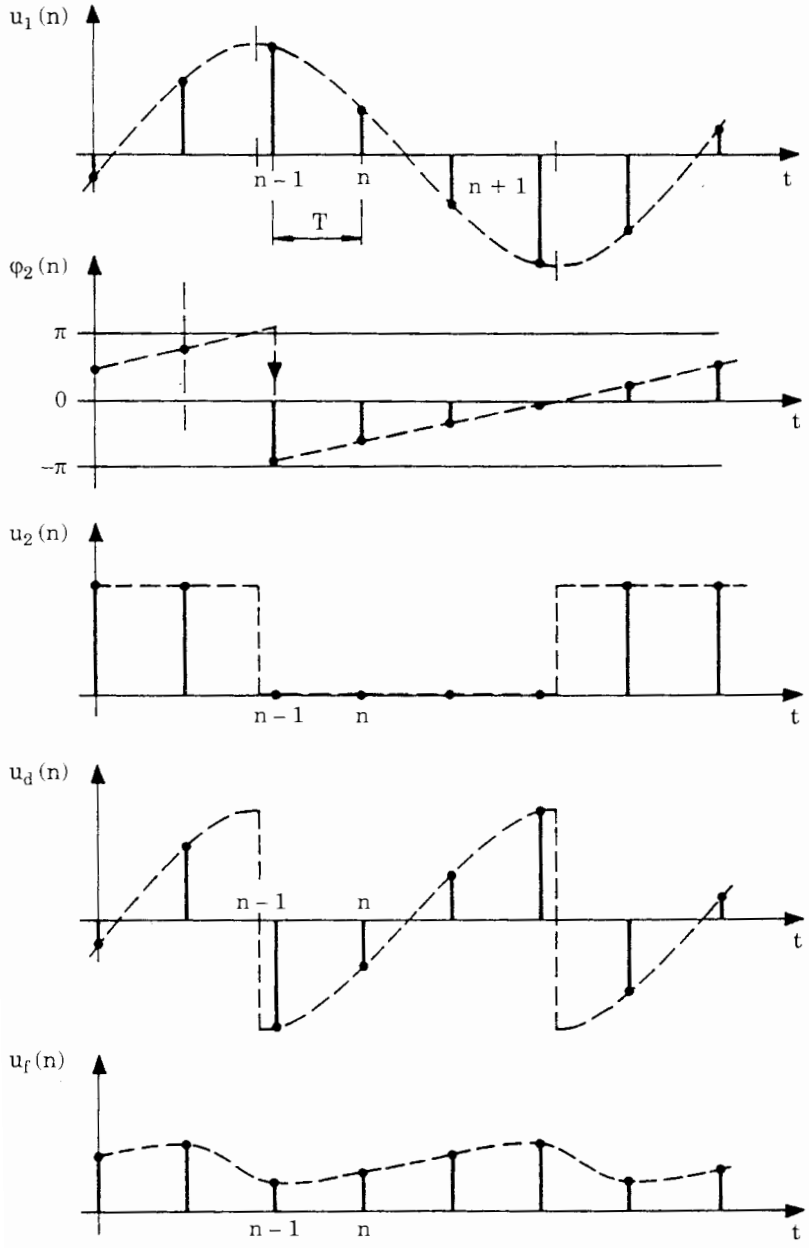


Figure 8.2 Plot of the signals that have to be calculated by the SPLL algorithm.

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 given below. 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. 8.3. When the algorithm is started, initial values are assigned to all relevant variables. The program enters an endless cycle thereafter; i.e., the algorithm within the box is repeatedly executed until the system is halted

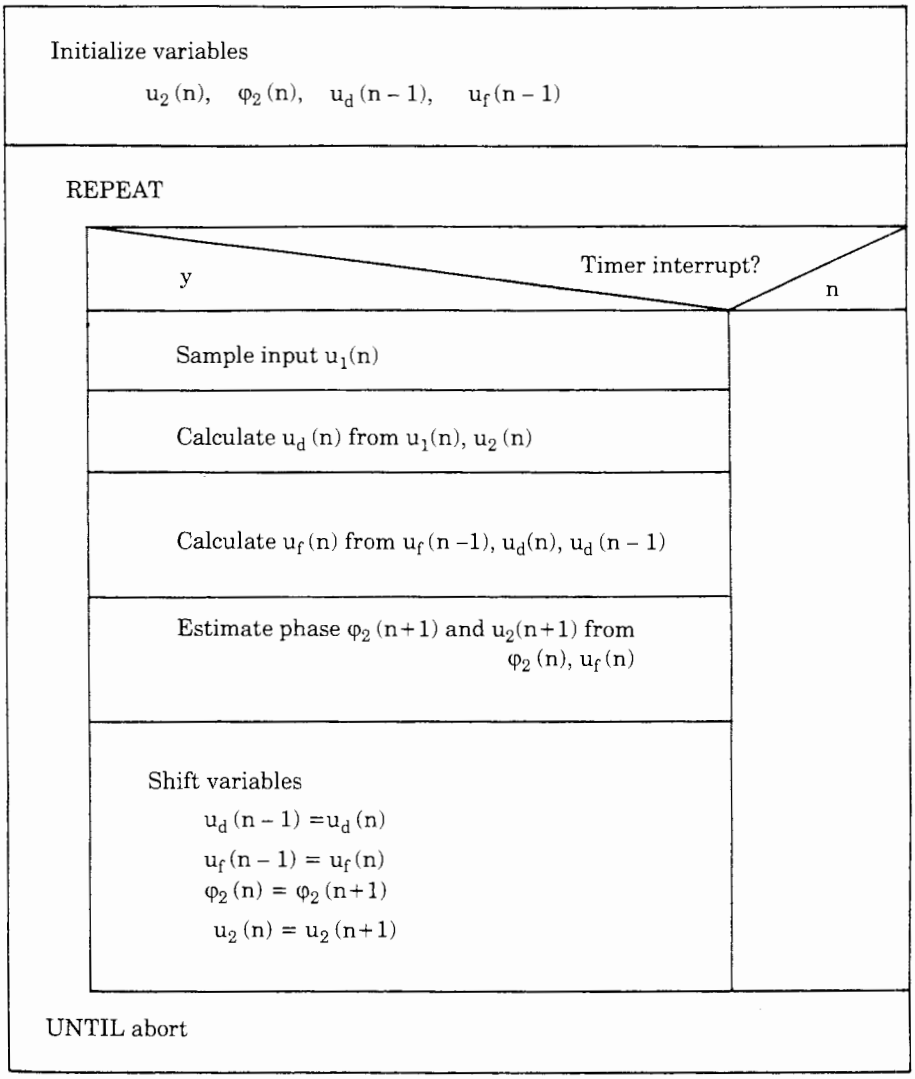


Figure 8.3 Structogram defining the operations of the SPLL according to Fig. 8.1.

or switched off. It is assumed that the clock signal (refer to Fig. 8.1) periodically generates interrupts in the microcontroller or whatever hardware is used. Thus interrupt requests show up at time instants $t = T, 2T, \ldots, nT$. 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. 8.4. 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:

$$u_f(n) = -a_1 u_f(n-1) + b_0 u_d(n) + b_1 u_d(n-1) \tag{8.5}$$

This is the recursion of a first-order digital filter. As pointed out in Sec. 6.1.2, an analog filter is described by its transfer function

$$F(s) = \frac{U_f(s)}{U_d(s)} \tag{8.6}$$

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:

$$F(z) = \frac{U_f(z)}{U_d(z)} \tag{8.7}$$

where z now is the z operator. There are a number of transforms[12,13,14,19] that can be used to convert $F(s)$ into $F(z)$. The most often used is called bilinear z transform. It will be covered in some more detail in Appendix 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 best PLL performance, we assume that $F(s)$ is the transfer function of the active PI filter; refer also to Eq. (2.28). Using the bilinear z transform, we get

$$F(z) = \frac{b_0 + b_1 z^{-1}}{1 + a_1 z^{-1}} \tag{8.8}$$

where the filter coefficients are given by

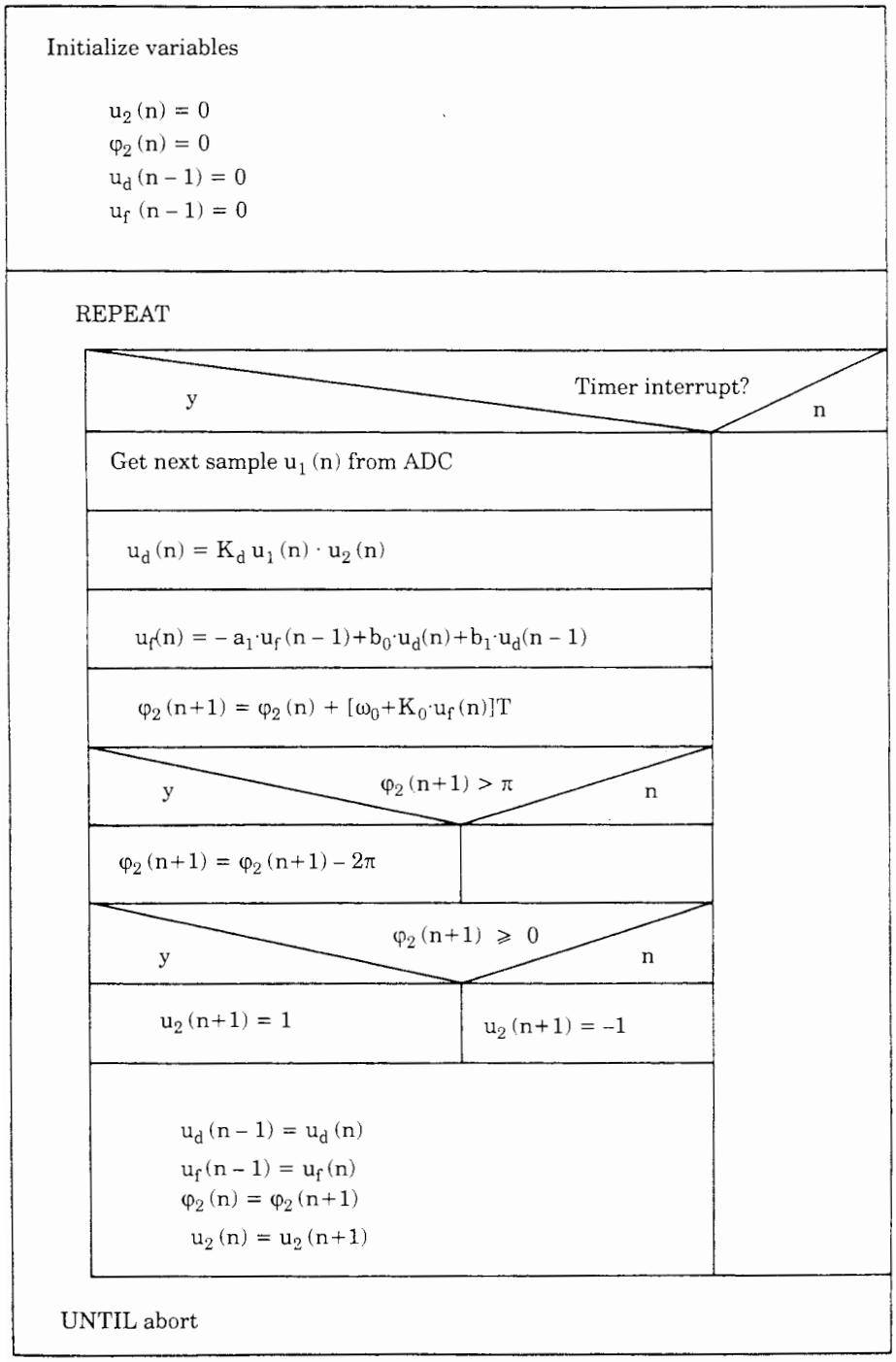


Figure 8.4 Structogram showing the algorithm to be performed by the SPLL in each sampling interval.

$$a_1 = -1$$

$$b_0 = \frac{T}{2\tau_1}\left[1 + \frac{1}{\tan(T/2\tau_2)}\right]$$

$$b_1 = \frac{T}{2\tau_1}\left[1 - \frac{1}{\tan(T/2\tau_2)}\right]$$

and T is the sampling interval. Transforming Eq. (8.8) back into time domain, we get the recursion

$$u_f(n) = -a_1 u_f(n-1) + b_0 u_d(n) + b_1 u_d(n-1) \tag{8.9}$$

which is also listed in the structogram of Fig. 8.4. From Eq. (8.4*b*) 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, φ_2 is limited to the range $-\pi < \varphi_2 < \pi$. Whenever the computed value of $\varphi_2(n + 1)$ exceeds π, 2π 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 $u_d(n)$, etc., are delayed by one sampling interval; i.e.,

$$u_d(n-1) \leftarrow u_d(n) \qquad \text{etc.}$$

As we stated in Sec. 5.2.5 when simulating the PLL on the PC, the sampling rate f_s for this SPLL algorithm must be chosen at least 4 times the reference frequency in order to avoid aliasing of signal spectra.

8.3.2 A DPLL-like SPLL

When the input signal u_1 of a PLL is a binary signal, it is more appropriate to implement an SPLL that performs like a DPLL. We develop an algorithm now performing 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. 8.5). It essentially consists of three functional blocks: a PFD algorithm, a digital filter, and a DCO. (No down-scaler is used in this configuration; that is, $N = 1$.) The digital filter is required to operate like the passive lead-lag filter in a DPLL, which is once more shown in Fig. 8.6. Before going into details of the last two figures, we consider the signals of this SPLL (Fig. 8.7). The only signal that physically exists—at least at the beginning—is the input signal $u_1(t)$, a square

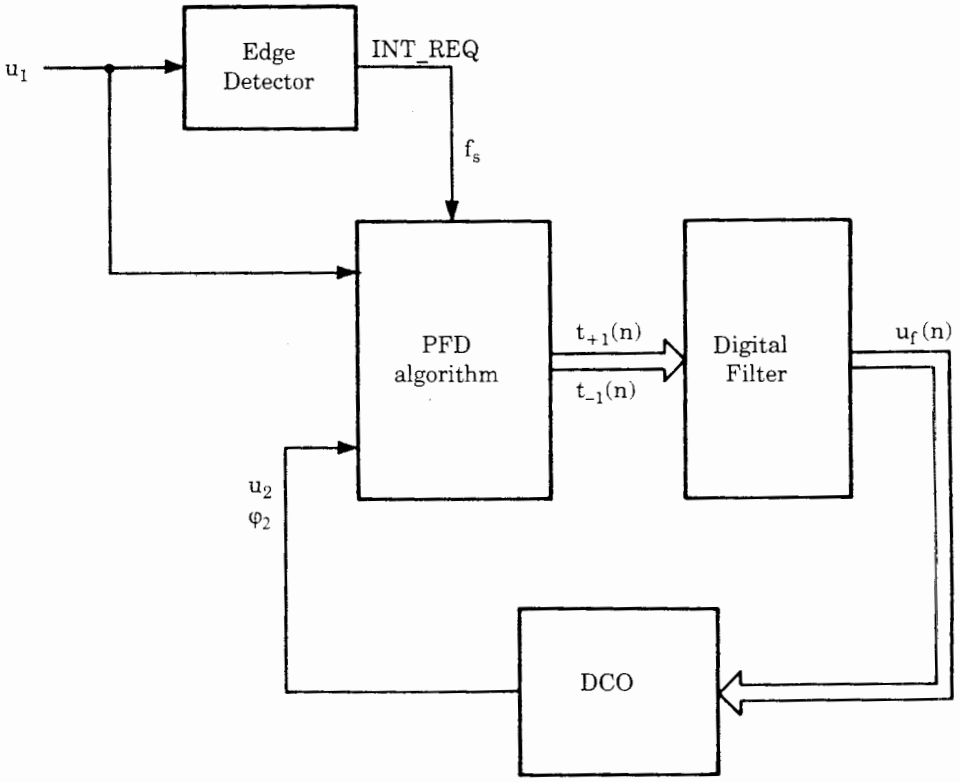


Figure 8.5 Block diagram showing the arithmetic and logic operations to be performed by an SPLL whose performance is similar to the DPLL.

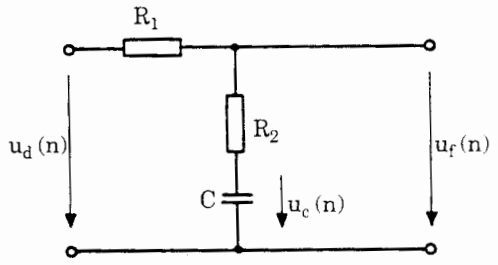


Figure 8.6 Schematic of the passive lag loop filter. This drawing is used to define the variables of the loop filter.

wave 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. 2.3.1, the logic state Q of the PFD depends on the positive edges of these two signals (or from the negative edges, whichever 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 time. Should the output frequency of the DCO drift away, the PFD

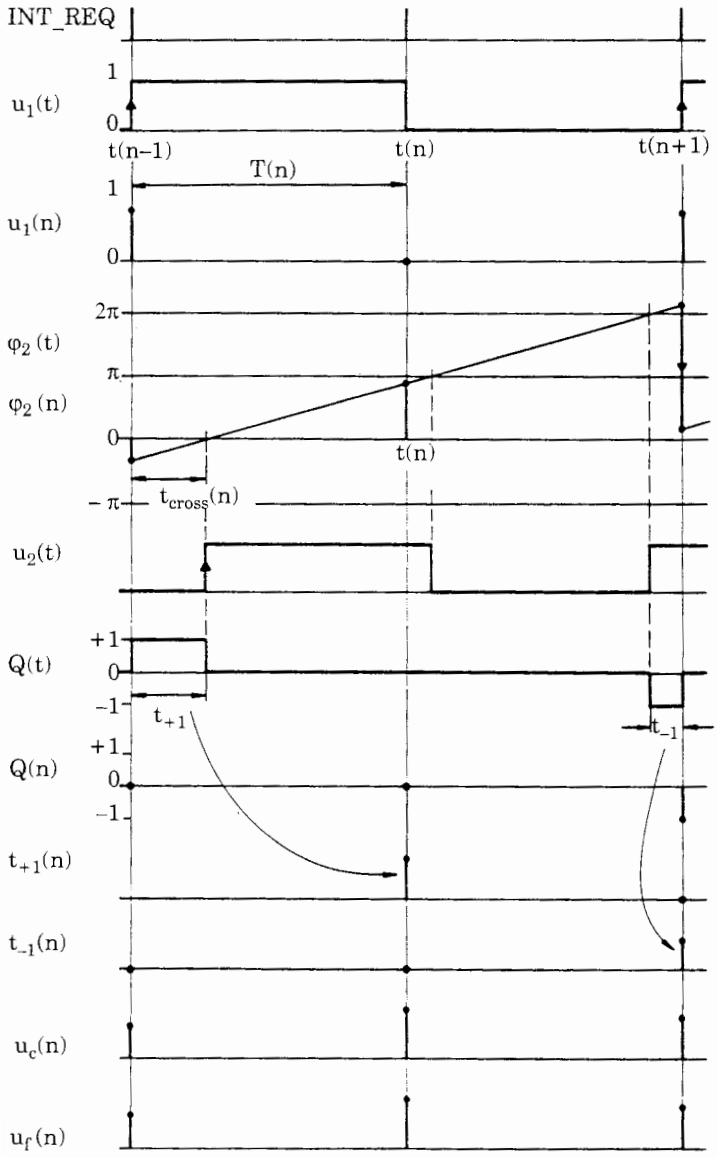


Figure 8.7 Plot of the signals that have to be calculated by the SPLL algorithm.

would generate correction pulses; that is, 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 be used, therefore, to detect the instants where the state of u_1 and u_2 is 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. 8.5 and 8.7. The computer is supposed to have a timer/counter chip such as the Intel 8253[32] or the AMD 9513.[33] As soon as the interrupt is recognized, a "time stamp" is taken; i.e., the time where the interrupt occurred is stored. The instants where interrupts have been detected are called $t(0), t(1), \ldots, t(n), \ldots$. Three of them are marked on top of Fig. 8.7.

Before the SPLL algorithm can be discussed, we have to define a number of signals (refer to Fig. 8.7):

$u_1(n)$ 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 of Sec. 8.3.1.

$Q(t)$ 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 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. 8.7. 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. 8.7.

$u_c(n)$ is used to denote the signal on the (fictive) capacitor C in the schematic of Fig. 8.6.

$u_f(n)$ is used to denote the sampled output signal of the digital filter in Fig. 8.5. With reference to Fig. 8.6, $u_f(n)$ 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. 8.8 shows what has to be done on every interrupt request. The signals appearing in the algorithm are shown in Fig. 8.7. 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 first waits for the next interrupt. When the interrupt has been detected, the time lapsed since the last interrupt is taken, $T(n) = t(n) - t(n-1)$. Next, the current value of the reference signal $u_1(t)$ is sampled, $u_1(n) = u_1(t)$. This is

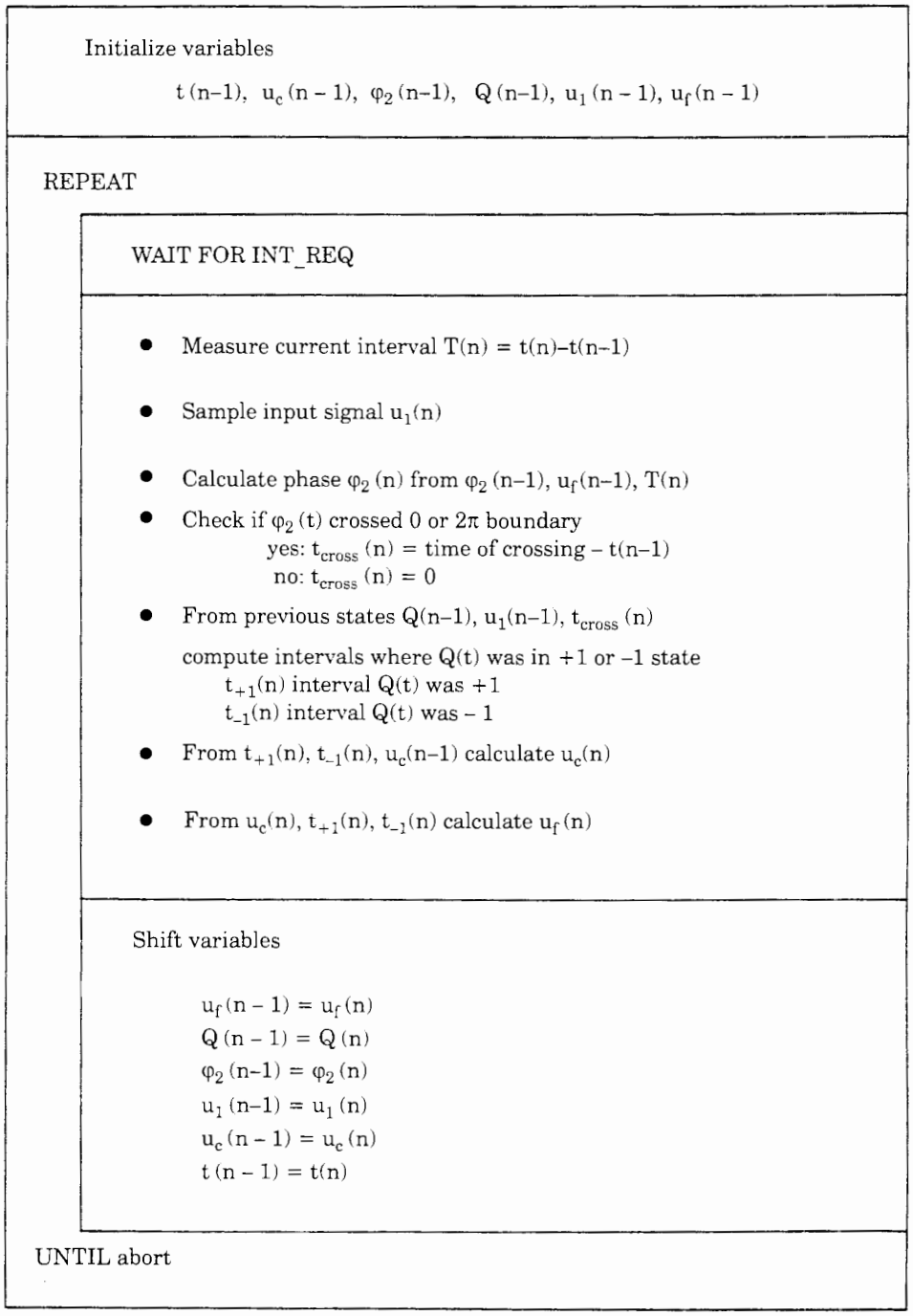
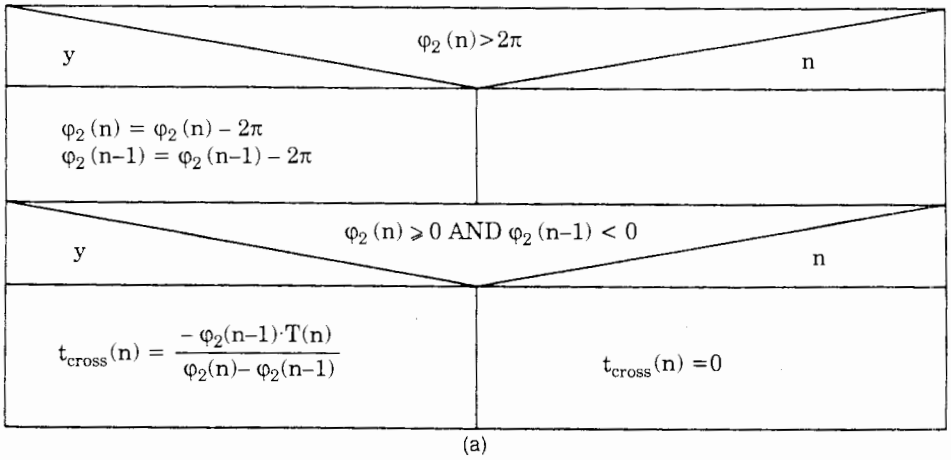


Figure 8.8 Structogram defining the arithmetic and logic operations within the SPLL of Fig. 8.5.

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. 8.7. 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 $u_f(n-1)$ could be calculated, and consequently we also knew 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

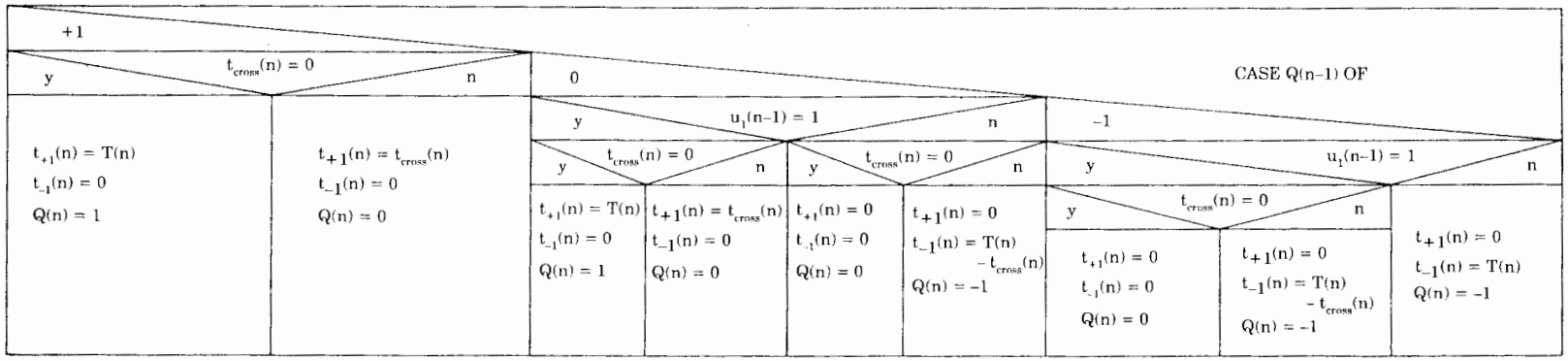
$$\varphi_2(n) = \phi_2(n-1) + [\omega_0 + K_0 u_f(n-1)]T(n) \tag{8.10}$$

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 2π during interval $T(n)$; this is sketched in the waveforms of Fig. 8.7. (The attentive reader will have noted that positive edges also would occur at phase crossing with 4π, 6π, . . . , etc. As will be shown later, we periodically reduce the total phase by 2π whenever it becomes larger than 2π. This is necessary to avoid arithmetic overflow in the computer.) When the phase crosses such a boundary, the corresponding time [i.e., the time interval from $t(n-1)$ to the crossing] is stored in the variable $t_{cross}(n)$. When no crossing is detected, $t_{cross}(n)$ is set to 0. The algorithm for the computation of $t_{cross}(n)$ is indicated in the structogram of Fig. 8.9*a*. It starts with the "normalization" of the phase signal $\varphi_2(t)$, as noted above.

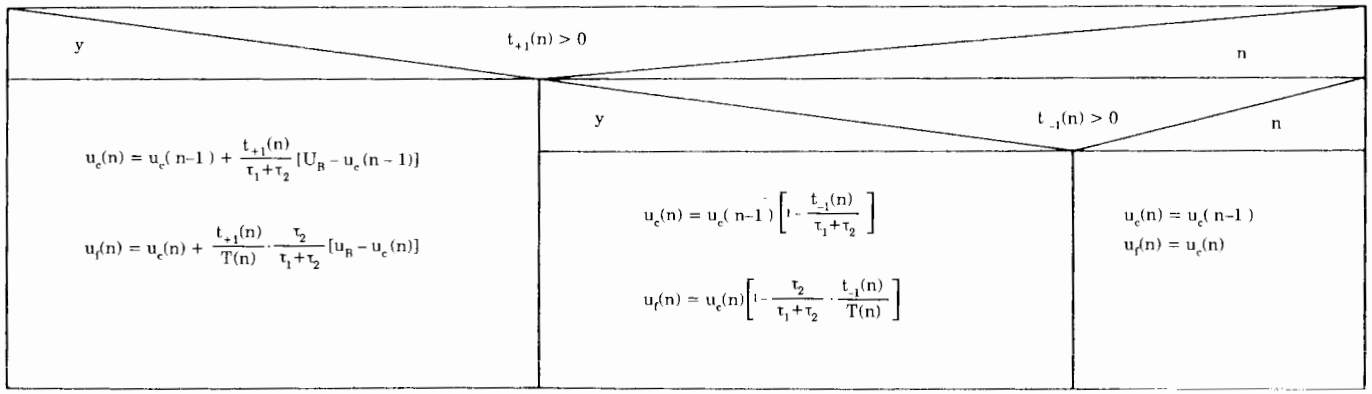


(a)

Figure 8.9 Detailed structograms of the algorithms to be performed by the SPLL of Fig. 8.5; (*a*) algorithm to determine the variable $t_{cross}(n)$ [time when output phase $\varphi_2(t)$ crosses boundary of 0 or 2π]; (*b*) algorithm for the PFD; (*c*) algorithm for the digital filter.



(b)



(c)

Figure 8.9 (*Continued*)

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. 8.7, $Q(n-1)$ 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, $Q(t)$ 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 $u_2(t)$. The behavior of the PFD is therefore case-sensitive; the algorithm in Fig. 8.9*b* 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 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. 8.6. 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. 8.9*c* 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. 8.6), 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 $t(n-1) \le t < t(n)$. This yields the expression listed in Fig. 8.9*c*. 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 τ_1 in Fig. 8.6. 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 the number of delayed samples of the output signal and from the 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 of Fig. 8.8. Finally, the structogram of Fig. 8.10 lists the full algorithm in mathematical statements. To avoid overloading the graph, the algorithms for $t_{cross}(n)$, for the PFD, and for the digital filter are shown separately (Fig. 8.9*a* to *c*). We note that the only signal that physically exists hitherto is the reference signal $u_1(t)$. A realistic SPLL system should also have one

Set all filter parameters
(τ_1, τ_2, U_B, ω_0, K_0 . . .)

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. 8.9a)

Algorithm for PFD (Fig. 8.9b)

Algorithm for digital filter (Fig. 8.9c)

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 8.10 Structogram showing the complete algorithm of the SPLL of Fig. 8.5.

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 has been computed. The situation would become more troublesome if the SPLL were requested to deliver the continuous DCO output signal $u_2(t)$. As we see from the waveforms in Fig. 8.7, the SPLL algorithm determines at time $t = t(n)$ when this signal made its last transition; i.e., it knows only what happened in the past. To output a real-time signal, the SPLL algorithm could extrapolate at time $t(n)$ how much time should elapse 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 has to be replaced by software.

8.3.3 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 would have to be replaced by interrupt requests, and the interrupt service routines would have to 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. 6.3) 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 rate of 9600 baud. This circuit would then operate with a center frequency of $f_0 = 9600\,\text{Hz}$. The K modulus of the K counter cannot be smaller than $K = 8$, as pointed out in Sec. 6.3. 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 $Mf_0 = 307{,}200\,\text{Hz}$. 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 $3\,\mu\text{s}$, including the operations required to service the interrupt request. By the present state of the art, this is out of the reach of simpler microcontrollers such as the popular 8051 family.[34] 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.

Chapter

9

The PLL in Communications

9.1 Types of Communications

In this chapter we will discuss applications of the PLL in the domain of data communications. There are many different kinds of communications, however; hence we will first look at the most important variants. First of all, 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[22] 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.[2]

9.1.1 From analog to digital

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 are 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 speak about 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 still are *amplitude modulation* (AM), *frequency modulation* (FM), and *phase modulation* (PM).

Historically, AM is the oldest modulation scheme. To modulate an analog signal (e.g., voice, music, and measurements of temperature or humidity) onto a carrier, an analog multiplier can be used. The multiplication can also be performed by a digital multiplier; this operation can also be executed by software on a suitable platform, e.g., on a microcontroller. If the multiplication is done

digitally, the modulated carrier signal must be converted into analog form by a DAC, of course. Demodulation of AM signals can be carried out by an LPLL circuit similar to that shown in Figure 2.47. When a linear PLL locks onto an AM carrier whose frequency is identical with the center frequency of the PLL, there is a phase difference of 90° between the carrier (u_1) and the VCO output signal (u_2). The 90° phase shifter in Figure 2.47 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 by a four-quadrant multiplier. A low-pass filter eliminates the unwanted high-frequency component at about twice the carrier frequency. Of course, the Schmitt trigger in Figure 2.47 would have to be removed from this circuit when it is applied as an AM demodulator. The bandwidth of the low-pass 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 low-pass 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; see 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 ω_2 is

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

and hence is proportional to the information signal u_f. The phase of the VCO output is the integral of ω_2 over time:

$$\varphi_2(t) = \omega_0 t + K_0 \int u_f(t)\,dt$$

which can be written as

$$\varphi_2(t) = \omega_0 t + \vartheta_2(t)$$

where $\varphi_2(t)$ represents the phase that carries the information. (The relationship between phase and radian frequency was explained in Sec. 2.2.) In PM, $\vartheta_2(t)$ 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, i.e., 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

$$F(s) = \frac{1}{1 + sT_i}$$

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 low-pass 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.

Baseband transmission of digital signals has already been discussed in Sec. 2.10.1, 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.

9.2 Digital Communications by Bandpass Modulation

Let us start with a review of the most important modulation schemes for digital communications.

9.2.1 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, a technique also called *on-off keying*. (In the first years 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 Figure 2.47, 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); see Table 10.1.

9.2.2 Phase shift keying

Phase shift keying (PSK) is the most widely used digital modulation today. Some of its descendants (QPSK, OQPSK, QAM) are more bandwidth efficient and will be discussed later. In classical PSK [also referred to as BPSK (binary PSK) or PSK_2] the polarity of a carrier is controlled by a binary data signal. Such a binary signal is shown in Fig. 9.1*a*, and the modulated carrier in Fig. 9.1*b*. 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 π (180°). The definition can be inverted if desired. As usual in the theory of alternating currents, amplitude and phase of the modulated carrier can be represented as a vector (phasor) in the complex plane, Fig. 9.1*c*. The length of the phasor represents its amplitude, the angle with the horizontal axis its phase. For BPSK the phase can only take two values, 0 and π.

As we will see in Sec. 9.4, there is a distinct relationship between the symbol rate R (number of binary symbols transmitted in 1 second) and the bandwidth W (in hertz) required to transmit the modulated signal. It turns out 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

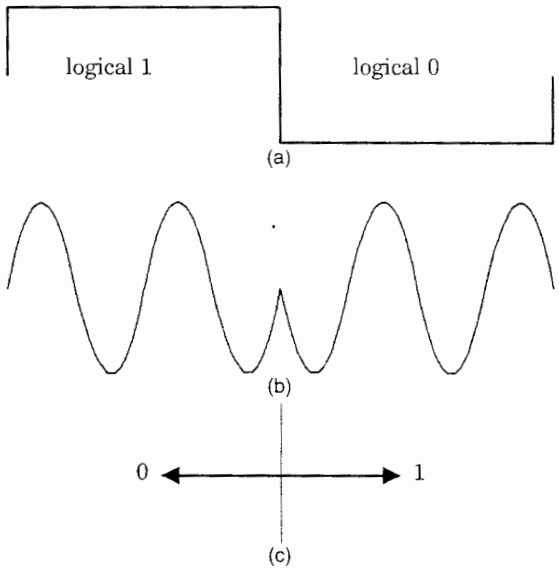


Figure 9.1 Binary phase shift keying (BPSK). (*a*) Binary signal. (*b*) Modulated carrier. (*c*) Vector plot of the modulated carrier signal.

(the throughput of binary symbols) without increasing the channel bandwidth. This seems contradictory at first glance, but we will see shortly that it is possible.

9.2.3 Quadrature phase shift keying

QPSK (*quadrature phase shift keying*, also called quaternary PSK or 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°. One of the carriers is called the *in-phase* component; it is assigned a phase of 0° and is mathematically represented by a *cosine wave*. The other carrier is called *quadrature carrier* and is assigned a phase of 90°. 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 makes sense only 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 or π (180^0), but the phase of the quadrature carrier can take on the values $\pi/2$ (90°) or $3\pi/2$ (270°). If we add the two carriers, the phase of the resulting signal can have values 45°, 135°, 225°, or 315°. Adding the two modulated carriers increases the symbol rate by a factor of 2, 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. 9.2. We note that with QPSK, 2 bits are transmitted in one symbol period.

Figure 9.2 explains the principle of QPSK. The binary data stream as delivered by the data source is shown in Fig. 9.2*a*. This sequence is partitioned into

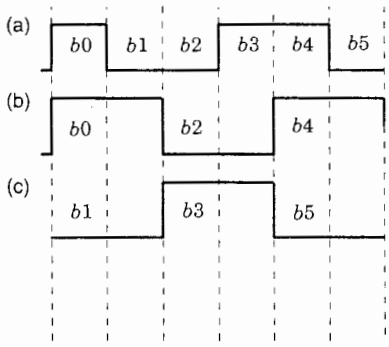


Figure 9.2 The principle of QPSK. (*a*) Binary signal to be transmitted. (*b*) Data stream corresponding to the even bits in *a*. (*c*) Data stream corresponding to the odd bits in *a*.

two data streams now: the even bits (b_0, b_2, etc.) form the binary sequence that modulates the in-phase carrier (Fig. 9.2*b*), while the odd bits (b_1, b_3, etc.) modulate the quadrature carrier (Fig. 9.2*c*). Figure 9.3*a* represents the four possible phases of the sum of in-phase and quadrature signals, and Fig. 9.3*c* shows the corresponding phasor plot.

As can be seen from Figure 9.2, both in-phase and quadrature data streams change their values at half the original symbol rate, i.e., on every second symbol delivered by the data source. In other words, the two data streams are *aligned*. In OQPSK (*offset* quadrature phase shift keying) the second data stream (Fig. 9.2*c*) is delayed by one symbol period of the source signal. In-phase and quadrature data streams no longer change their states simultaneously, but are staggered in time. This does not have any impact on channel capacity or bandwidth, but rather offers some practical benefits. When the data streams are low-pass filtered (as will be explained in Sec. 9.4), the data signals are no longer square waves, but become smoothed. Whenever a data signal changes polarity then, the modulated carrier temporarily fades away; hence we temporarily loose the signal that the receiver tries to lock onto. This drawback becomes less severe if both of the data streams cannot change state at the same time. This is the motivation to use OQPSK.

9.2.4 QAM (*m*-ary phase shift keying)

Having increased the channel capacity by a factor of 2 without sacrificing bandwidth, we are going now to 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, say 1 or 2? When we do so, the modulated in-phase carrier could take on

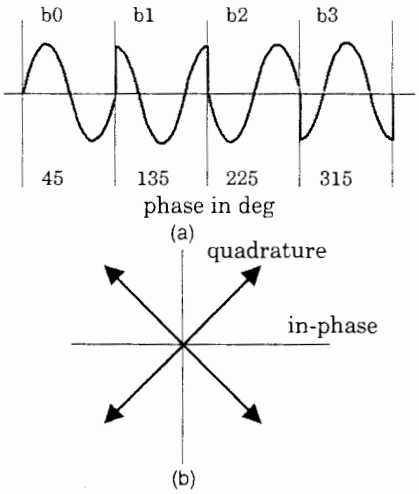


Figure 9.3 Phase relationship of BPSK signals: (*a*) the modulated carrier can take on four different phases; (*b*) phasor diagram of modulated carrier.

a phase of 0° with an amplitude of 1, or a phase of 0° with an amplitude of 2, or a phase of 180° with an amplitude of 1, or a phase of 180° 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 is also in a 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 4 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 2 values—more values can be used. A phasor diagram of QAM can look like Fig. 9.4. Here two amplitude values are used. The end points of the phasors have identical distances on the horizontal and vertical axes. 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. With the scheme shown in Fig. 9.4, the combined carrier can take on 16 states, and hence 4 bits per symbol are transmitted. The code shown in Fig. 9.4 is commonly called QAM_{16}. Note that QAM_4 is identical with QPSK. Theoretically the set of amplitudes can be made as large as desired, but noise superimposed on the transmitted signal sets limits. To discriminate between different amplitude levels, the receiver must define *decision limits*. With reference to Fig. 9.4, 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 by using statistics and Shannon's information theory.[20] Today QAM is the most widely used technique. In modern modems, QAM_{64}, QAM_{128}, and QAM_{256} come into play. As we will see in Sec. 9.6, QAM will be used very efficiently in digital video broadcasting (DVB).

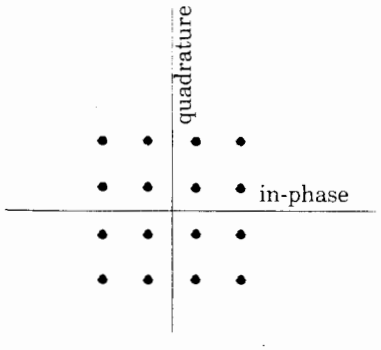


Figure 9.4 Phasor diagram of QAM.

9.2.5 Frequency shift keying

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 us discuss baseband communication first. In its simplest form (FSK_2), two different frequencies f_1 and f_2 are defined, where f_1 represents a binary 0, and f_2 represents a binary 1 or vice versa, Fig. 9.5*a*. The two frequencies can be chosen arbitrarily, for example, $f_1 = 1650\,\text{Hz}$, $f_2 = 1850\,\text{Hz}$. 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 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*, Fig. 9.5*b*. Here the frequencies f_1 and f_2 are chosen as to fulfill the relationship

$$f_i = k_i \frac{1}{2T_s} \tag{9.1}$$

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. 9.5*b*, $k_1 = 4$ 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

$$\begin{aligned} \int_0^{T_s} s_i(t)s_j(t)\,dt &= 1 \qquad i = j \\ &= 0 \qquad i \neq j \end{aligned} \tag{9.2}$$

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, 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.

We considered only 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. In doing so, 2 bits per symbol period can be transmitted. There are a number of simple modems that use FSK_2 or FSK_4 and work immediately with baseband signals. Of course, the baseband signal can be modulated onto a high-frequency carrier. When this is done, many FSK-modulated carriers (with different carrier frequencies, of course) can share one single link.

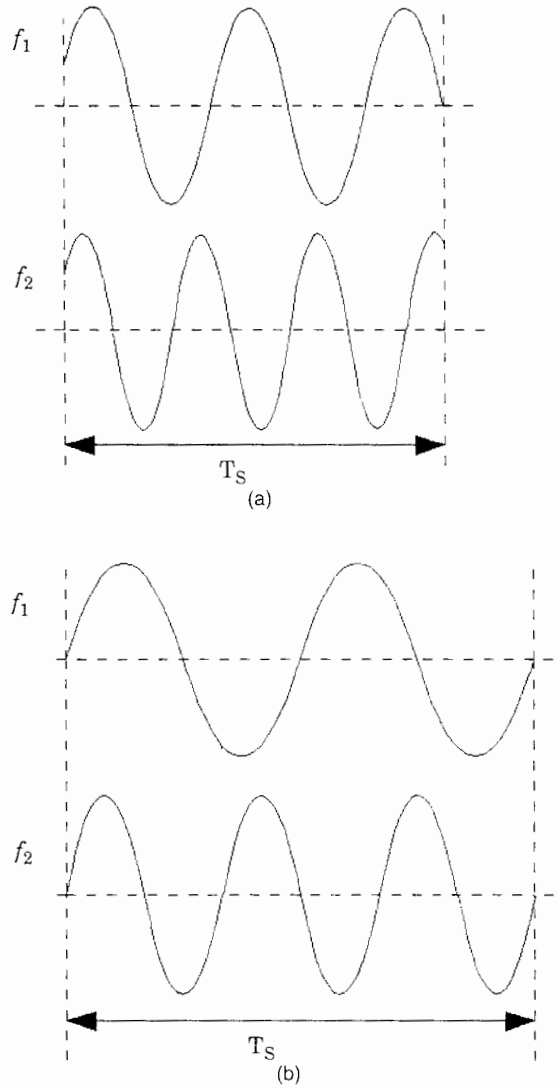


Figure 9.5 Definition of binary FSK: (*a*) Nonorthogonal FSK: arbitrary values for f_1 and f_2 are chosen. (*b*) Orthogonal FSK: there is a tight relationship between frequencies f_1 and f_2; see text.

In the case of nonorthogonal FSK, switching abruptly from one frequency to another can cause sharp transients, which extends the bandwidth of the signal in an undesired manner. This drawback is avoided in a special version of FSK called *MSK* (*minimum shift keying*).[20,49] With MSK, every symbol waveform starts and ends with a zero crossing of the same direction, e.g., from negative to positive values (as can be seen from Fig. 9.5*b*). Adjacent symbol waveforms then never execute phase steps. This reduces the bandwidth required to transmit the FSK signal.

9.3 The Role of Synchronization in Digital Communications

Before discussing the various modulation techniques applied in digital communications, 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 has to be recovered by demodulation. In most cases, the demodulation is performed synchronously; i.e., the receiver generates a replica of the carrier and multiplies the incoming signal with that reconstructed carrier. This shifts the spectrum of the received signal by a frequency offset that is equal to the carrier frequency, and the data signal is obtained by simple low-pass filtering. Because the replica of the carrier must be in phase with the latter, *phase synchronization* is required. 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 (e.g., 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 that 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, 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 in blocks. Such a block may consist of a block header, followed by the actual data and eventually by a trailer that contains check code [e.g., 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.[20]

9.4 Digital Communications Using BPSK

9.4.1 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 by Fig. 9.1, 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 is its very steep transients. Normally the NRZ format is used to modulate the carrier (see Fig. 2.50). 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.

As Nyquist realized in 1928,[40] 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 low-pass-filter the data signal at the transmitter. The most appropriate filter would be a "brickwall filter," i.e., an ideal low-pass filter having gain 1 at frequencies below half the symbol rate and gain 0 elsewhere (Fig. 9.6, see curve label $r = 0$). The impulse response of the brickwall filter is

$$h(t) = \frac{\sin(\pi t/T)}{\pi t/T} \tag{9.3}$$

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. 9.7 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 finite impulse response (FIR) digital filter (see App. C) to any desired level of accuracy. However, because the impulse response decays slowly to 0, this leads to excessively long FIR filters, i.e., 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.

Every FIR filter causes the signal that is passed to be delayed, where the delay τ is given by

$$\tau = \frac{L-1}{2} T_F \tag{9.4}$$

L is the length of the FIR filter and T_F is the sampling interval of the filter. In Fig. 9.7 the delay of the filter has been neglected; hence its maximum (+1) occurs at time $t = 0$. Normally such a filter would be sampled at a frequency f_F that is an integer multiple of the symbol rate $R = 1/T$, where T is the symbol period. If we used an FIR filter of length 33, for example, it would have a delay of 16 T_F; if we assume furthermore that the filter sampling frequency f_F is 4 times the symbol rate R, the filter would delay the signal by 4 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.

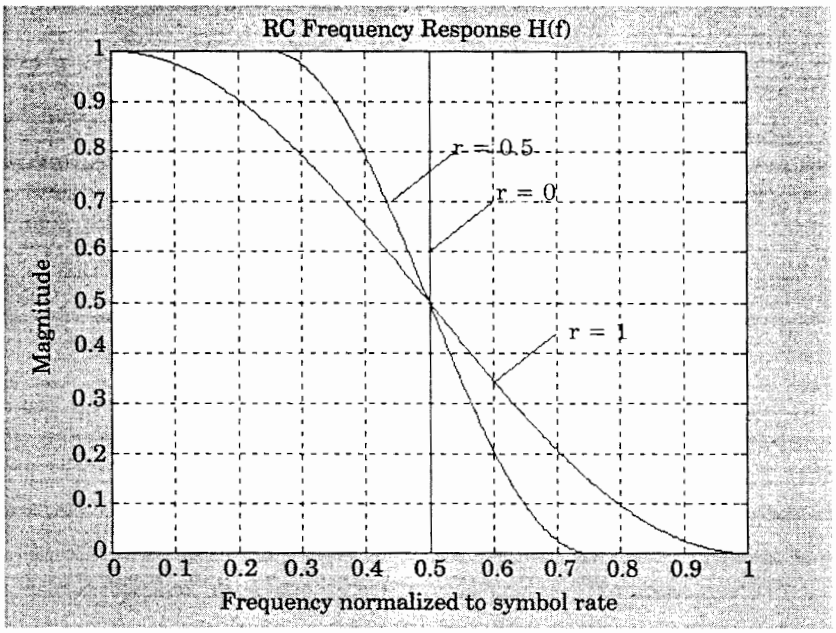


Figure 9.6 Amplitude response of brickwall filter ($r = 0$) and raised cosine filter ($r > 0$).

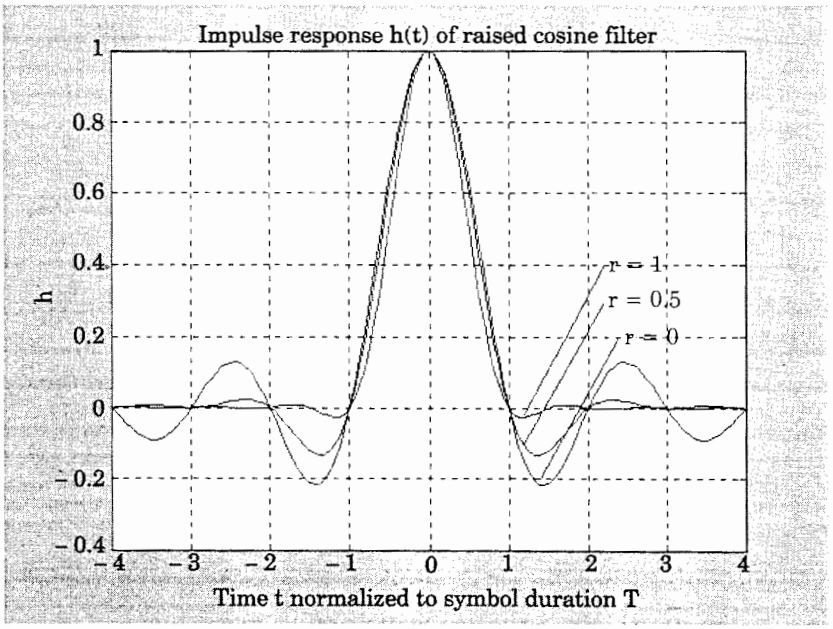


Figure 9.7 Impulse response of the raised cosine filter for various values of (relative) excess bandwidth r.

9.7; i.e., 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 $t = T, 2T, 3T$, etc. Now we remember 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 imput. 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 are passed through the filter in succession. If two succeding 1s are supplied, the first logical 1 will be represented by a delta function with amplitude +1 applied at $t = 0$, and the second logical 1 by another delta function with amplitude +1 applied at $t = T$.

This situation has been plotted in Fig. 9.8. The upper trace shows the two delta functions (marked by arrows), the lower trace the corresponding impulse responses (solid curves). The zeros 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 would have to sample the filtered signal exactly at time $t = 0, T, 2T$, etc. As can be seen from Fig. 9.8, the amplitude of the combined signal at $t = 0$ (marked by a square) depends uniquely on the impulse applied at $t = 0$; the impulse response caused by the

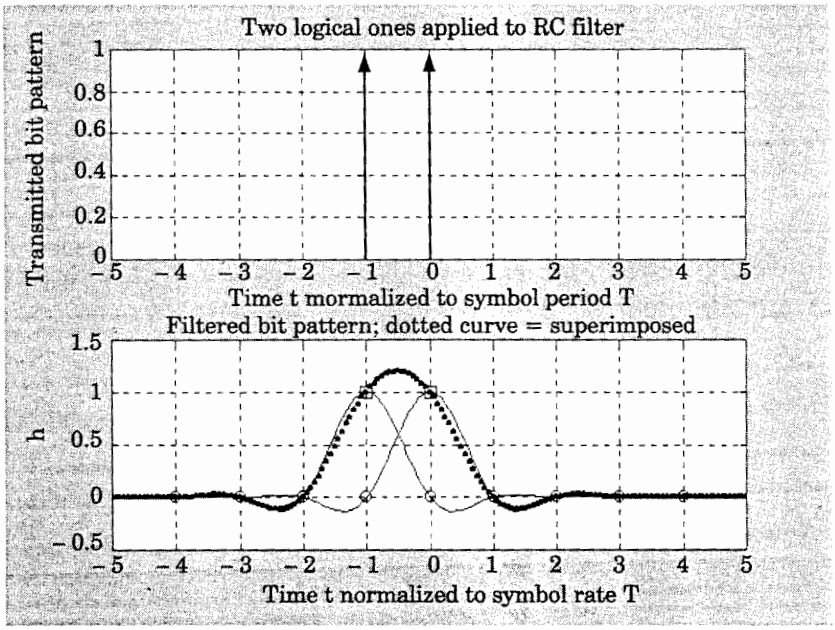


Figure 9.8 Response of the raised cosine filter to a sequence of two binary 1s applied to its input. Upper trace: a sequence of two delta functions corresponding to two binary 1s transmitted in succession. Lower trace: the solid curve (thin) whose maximum is at $t/T = -1$ is the response of the filter 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.

second delta function is zero at this time. The same holds for the combined signal at $t = T$. Its amplitude at $t = T$ (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*; when the brickwall filter is used, there will be no *intersymbol 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 low-pass filter. Its impulse response will be a damped oscillation. It also passes through zero, but these zeros are not at $t = 0$, T, $2T$, etc.; hence the impulse responses coming from delta functions applied to the input will interfere, or in other words, 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 zeros 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, i.e., 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. 9.6 for three values of 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

$$H(f) = \begin{cases} 1 & \text{for } |f| < \dfrac{1-r}{2T} \\ \cos^2 \dfrac{\pi}{4} \dfrac{2T|f| + r - 1}{r} & \text{for } \dfrac{1-r}{2T} \le |f| \le \dfrac{1+r}{2T} \\ 0 & \text{for } |f| > \dfrac{1+r}{2T} \end{cases} \tag{9.5}$$

Figure 9.7 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 0.15 to 0.35 are customary. For completeness we also give the expression for the impulse response $h(t)$ of the RCF[41]:

$$h(t) = \frac{\sin(\pi t/T) \cdot \cos(\pi r t/T)}{(\pi t/T)\left[1 - \left(\frac{2rt}{T}\right)^2\right]} \tag{9.6}$$

Designing of an FIR raised cosine filter is an easy task: Eq. (9.6) 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 singularity is removed by replacing both numerator and denominator by their derivatives; mathematically this is called L'Hôpital's rule. The computation is made even easier if Matlab's Signal Processing Toolbox is available.[25] 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 low-pass filtering because it completely suppresses ISI. As we will see in Sec. 9.4.2, the receiver also needs a low-pass filter, for different reasons. It turns out that we cannot use another raised cosine filter at the receiver; if we did, the data signal would have to 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 so-called root raised cosine filter (RRCF) will fix that problem.

9.4.2 Receiver considerations

A number of tasks have to 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 propagates across the data link. When coherent detection has to be applied, the receiver must know the phase of the carrier exactly; i.e., the receiver must reconstruct a replica of the carrier. As we will see in the following, this is more difficult than it appears at a first glance. If noncoherent detection is to be realized, the receiver must not know the phase of the carrier. Instead, it 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. 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; i.e., 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. Because of its higher noise immunity, coherent detec-

tion is the preferred method in most receiver circuits today,[42] so we will concentrate on this technique.

Phase synchronization within the receiver. In 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, ranging 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, 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 simply means that 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.[1,20]

The squaring loop. One of the earliest circuits is the *squaring loop*, Fig. 9.9. The radian frequency of the carrier is ω_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 and −1. 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

$$s^2(t) = m^2(t)\sin^2(\omega_1 t)$$

According to the multiplication theorems of trigonometry, this consists of a dc term (which can be discarded here) and a high-frequency term of the form $\cos(2\omega_1 t)$, that is, a term having twice the frequency of the carrier. A conventional PLL is used then 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 by analog circuitry, alternatives have been searched for. A very successful solution is the *Costas loop*.[1,20] It originally was realized as an analog circuit, but is implemented mostly by digital techniques today. An example is the digital Costas loop HSP50210 manufactured by Harris.[42] The basic Costas loop is shown in Fig. 9.10. 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

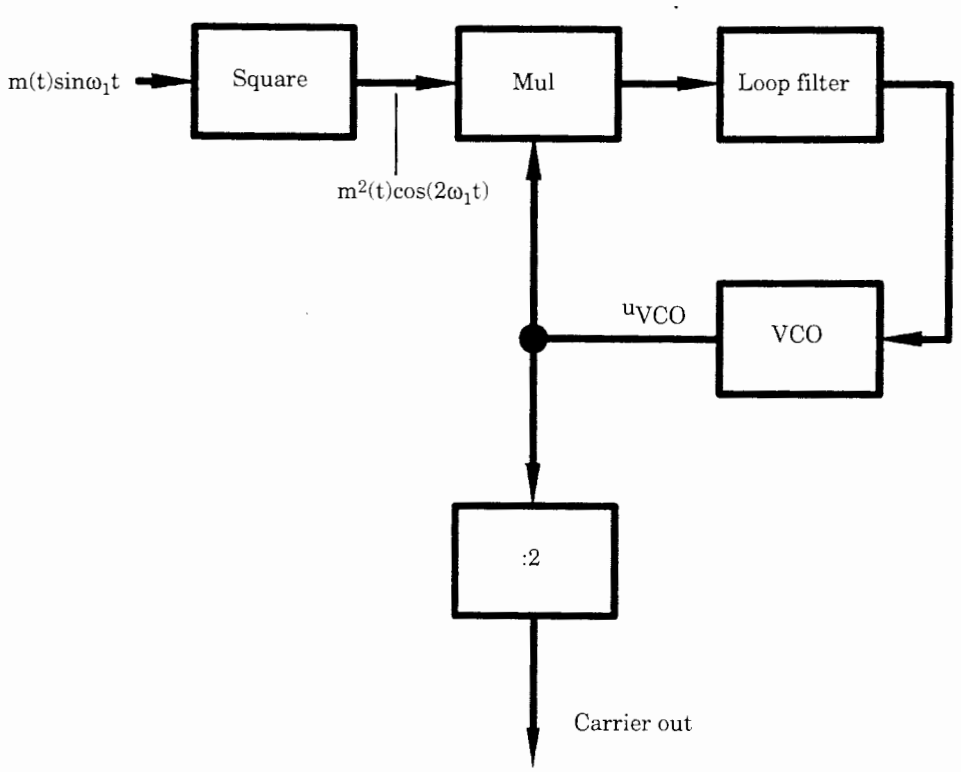


Figure 9.9 Squaring loop.

carrier (the output signal of the VCO) by $\sin(\omega_1 t + \theta_2)$. In the locked state, the phases θ_1 and θ_2 will be nearly the same.

Let us 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; i.e., connect the output of the loop filter in the lower branch directly to the input of the center low-pass filter. This forms a rather conventional PLL. The multiplier at 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°, as usual with 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 inverted. To overcome the problem, the I branch is added, together with the additional multiplier at center right. As indicated in Fig. 9.10, the output signal of the loop filter in the I branch is given by $m(t)$

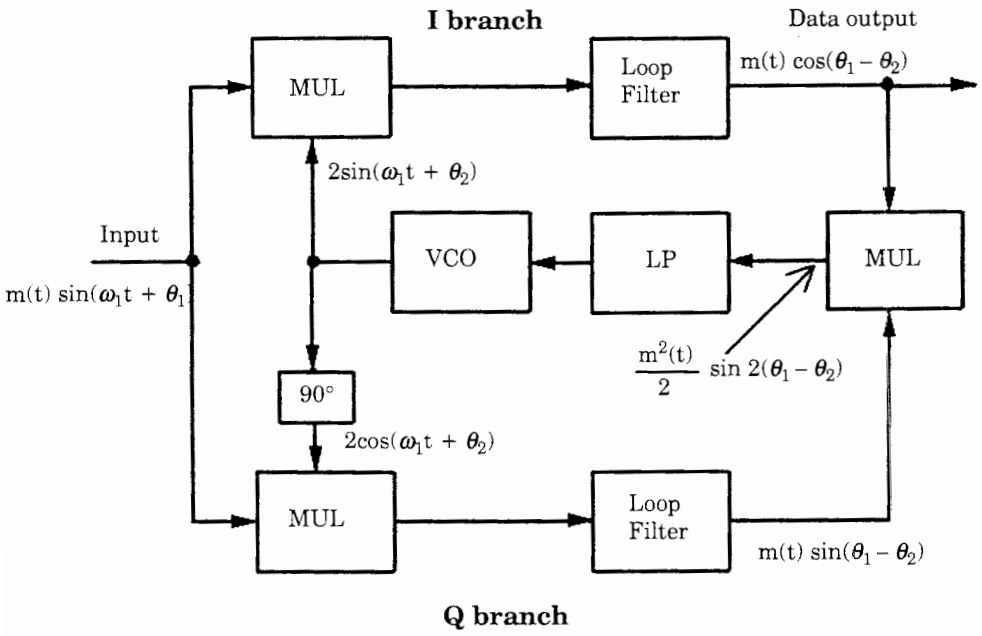


Figure 9.10 Costas Loop for BPSK.

$\cos(\theta_1 - \theta_2)$. If phase is nearly correct for tracking ($\theta_1 - \theta_2$ 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 branches) 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 θ_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 recognize from Fig. 9.10, 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 low-pass-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 just check one single sample of the received symbol, but instead would form some statistic (e.g., 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*.[20]

What type of correlation filter do we need? There are many variants of the correlation filter. First of all, the correlation can be performed with the undemodulated or demodulated symbol waveform. Correlating the undemodulated waveform is simpler to explain; hence we start with this topic. Figure 9.11 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 π (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 that the sampling rate is 4 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 an FIR filter having length 4. To be more general, assume that the 4 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 reversed 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), etc. At this instant the upper filter outputs a 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 symbol sequence (with time delay $\tau = 0$). We now consider what happens with the lower FIR filter. Its coefficients are defined by [$-d$ $-c$ $-b$ $-a$] and 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

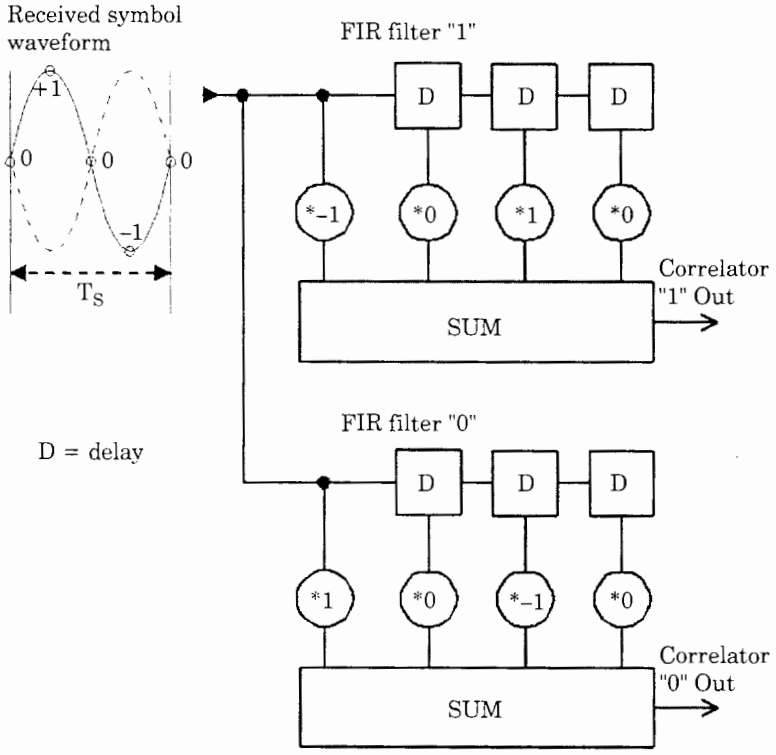


Figure 9.11 Block diagram of a correlation receiver. The incoming symbol is correlated with the template functions that are stored in the receiver.

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 that this error does not lead to a wrong decision. An error occurs only if the noise level is so high that it reverses the polarities of the correlator output signals.

In this example the waveform for logical 0 was chosen 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. 9.11 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 reversed order. It can be shown that the decision made by a correlation receiver is a *maximum likelihood* decision in the statistical sense,[20] 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) to the received symbol.

What about raised cosine filters? As stated, the described correlation method works best with carriers that are modulated with a square wave data signal. To reduce the bandwidth, modern digital communications mostly work with filtered data, as discussed in Sec. 9.4.1. When considering the working principle of the Costas loop (refer again to Fig. 9.10), 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 intersymbol interference (ISI) can occur. The raised cosine filter has been considered an optimum solution for this problem. But now in the Costas loop the data signal passes through an additional low-pass filter. As was demonstrated by the correlation receiver in the example of Fig. 9.11, the qualitiy of symbol decision in the receiver would be optimum when the same RCF is used 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 be no longer symmetrical about half the symbol rate, and ISI would be generated. The frequency response of the low-pass 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 low-pass filter. In the best case, we could try to approximate a brickwall filter, but this is suboptimal and results in an FIR filter with a very great number of taps. Moreover the brickwall filter would not be matched to the symbol waveform, and hence 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

$$H(f) = \begin{cases} 1 & \text{for} \quad |f| < \dfrac{1-r}{2T} \\ \cos\dfrac{\pi}{4}\dfrac{2T|f|+r-1}{r} & \text{for} \quad \dfrac{1-r}{2T} \le |f| \le \dfrac{1+r}{2T} \\ 0 & \text{for} \quad |f| > \dfrac{1+r}{2T} \end{cases} \tag{9.7}$$

The impulse response $h(t)$ of the RRCF is given by[41]

$$h(t) = \frac{\sin\left[\dfrac{\pi t}{T}(1-r)\right] + \dfrac{4rt}{T}\cos\left[\dfrac{\pi t}{T}(1+r)\right]}{\dfrac{\pi t}{T}\left[1-\left(\dfrac{4rt}{T}\right)^2\right]} \tag{9.8}$$

Like the impulse response of the RCF, the impulse response given by Eq. (9.8) 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 numerator and denominator by their derivatives.

The frequency response of the RRCF is no longer symmetrical about 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 to that of the RCF; hence 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 else 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 shown in Fig. 9.12. This schematic can be considered part of the block diagram of the Costas loop in Fig. 9.10. 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 that is an integer multiple of the symbol rate f_S. (In Fig. 9.12, $f_C = 1/T_C$ and $f_S = 1/T_S$.) The output of the matched 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. 9.8. To reconstruct the data correctly, it is very important that the receiver sample the matched filter ouput at the correct times, i.e., precisely at $t = 0, T, 2T, \ldots$. 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; these will be discussed at the end of this section.

The "integrate and dump" circuit. When the data signal is not low-pass-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

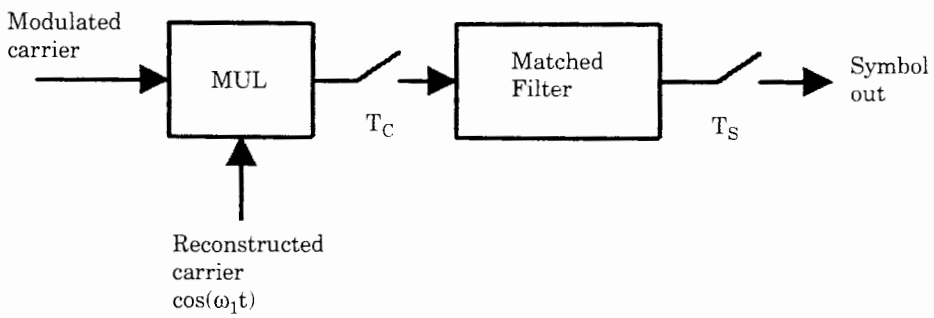


Figure 9.12 Principle of a matched filter that correlates the demodulated symbol waveform with a template stored in the receiver.

during the symbol period (+1 or −1). Under this condition, the correlator can be built from an integrator, as shown in Figure 9.13*a*. At the end of a 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 (Fig. 9.13*b*). The staircase function reaches its maximum at the end of the symbol. If the incoming symbol were 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 an *integrate and dump circuit.* Integrated digital Costas loops such as the Harris HAS50210[42] provide both RRCF and integrate and dump circuits. They can be configured individually according to the 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 that is locked in frequency and phase to the

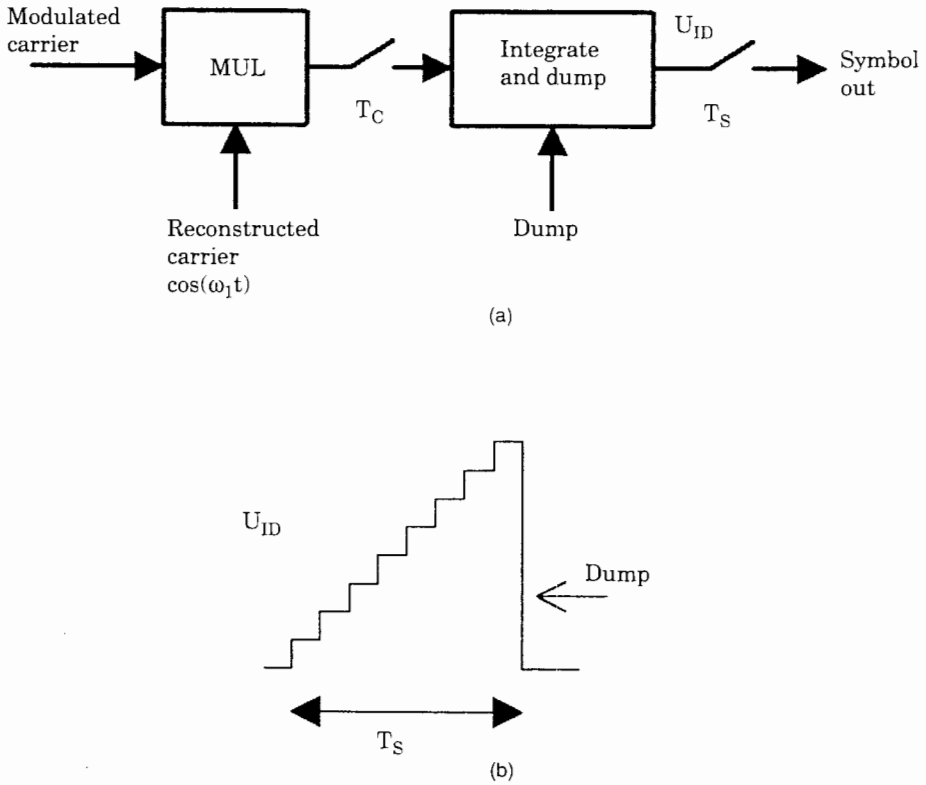


Figure 9.13 Integrate and dump circuit: (*a*) block diagram; (*b*) output of the integrator for a square wave input.

incoming carrier. When synchronizing with symbol timing, the receiver must recreate the symbol clock in order to know exactly when the end of a symbol period has been reached. We remember that the receiver has to sample the output of the correlation filter at the instant where the incoming symbol is terminated and the next symbol period is starting. There are many methods to achieve symbol synchronization; we discuss only the most often used.

"Midsymbol" sampling. We first discuss a method called *midsymbol synchronization*. It is mainly used when the data are low-pass filtered, e.g., by the root raised cosine filter. The correlator output is then sampled at the end of each symbol period (which corresponds to the instants $t = -2T, -T, 0, T, 2T$, etc., in Fig. 9.8). If a sequence of 1s is received, the end-symbol amplitudes will all be around +1. If a sequence of 0s is transmitted, all succeding 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. 9.14. When the symbol clock is properly adjusted, the zero crossing will be in the center between two succeding decision points (labeled *End symbol "1"* and *End symbol "0"* in the figure). The expected midsymbol 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 [the point labeled *Mid symbol (actual)*], the symbol timing is erroneous and must be corrected. The timing error (the difference between actual and expected location of midsymbol) is used to adjust the frequency of a local symbol clock generator. The midsymbol 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*, Fig. 9.15. This method works best when the data are square wave signals. In this case the symbol signal $s(t)$ is constant and

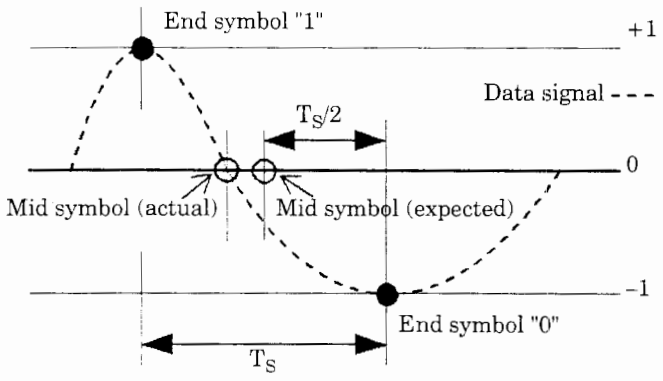


Figure 9.14 Symbol synchronization using midsymbol values.

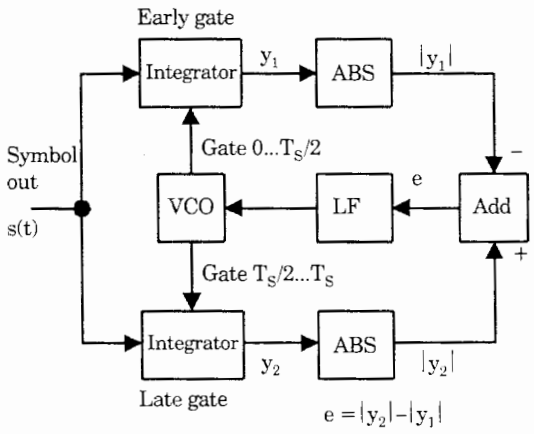


Figure 9.15 Symbol synchronization using the early-late gate technique.

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. 9.16*a* (the shaded areas). The output signals y_1 and y_2 are equal then, and the error signal e in Fig. 9.15 is zero. When there is an offset in symbol timing, the outputs of the integrators become unequal (Fig. 9.16*b*), and the error signal becomes positive or negative. Its polarity depends on the sign of the timing error. The waveforms in Fig. 9.16 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. 9.15).

Additional symbol synchronization circuits have been discussed in references.[1,20]

9.5 Digital Communications Using QPSK

9.5.1 Transmitter considerations

QPSK has been defined in Sec. 9.2; refer also to Fig. 9.3. To implement QPSK, two carriers have to be generated in the transmitter that are offset in phase by 90°. As explained by Fig. 9.2, the symbol stream is partitioned into two data streams; one of those 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. 9.17. This is a block diagram of the Harris HSP50307 burst QPSK modulator.[43] 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

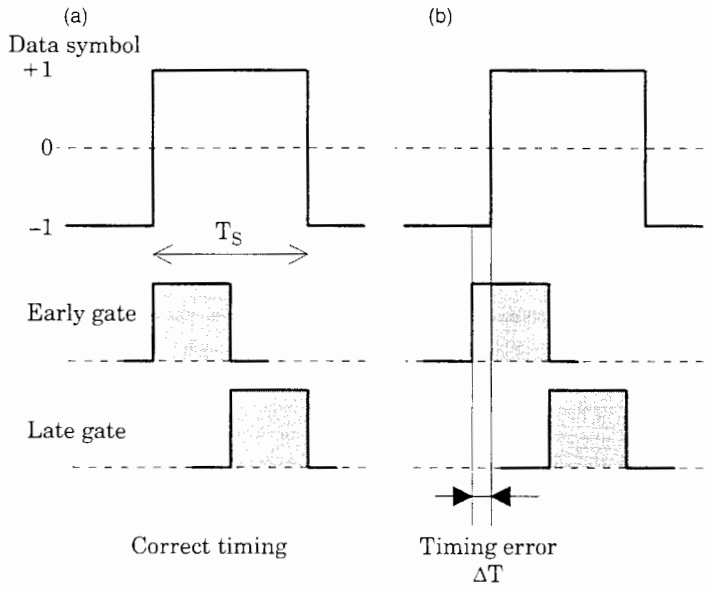


Figure 9.16 Gate timing of the early-late gate circuit. (*a*) Waveforms for correct timing. (*b*) Waveforms for timing error ΔT.

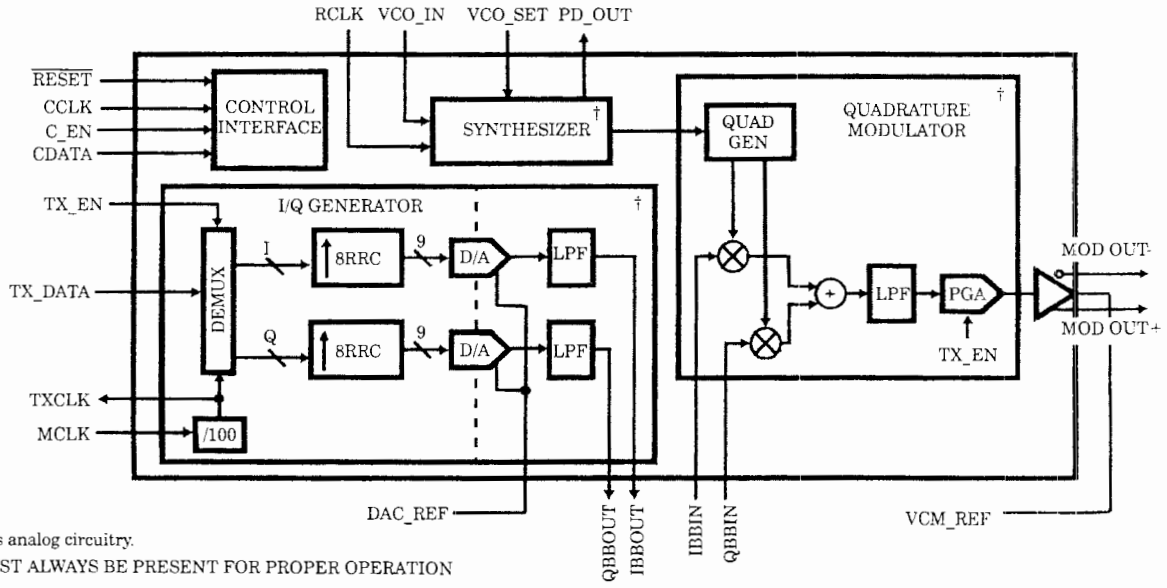


Figure 9.17 Block diagram of burst QPSK modulator Harris HSP50307. (*Used with the permission of the Harris Corporation.*)

filter has order 64 (length 65) and is clocked at a frequency of 1.024 MHz; that is, the filter input signal is oversampled by a factor of 8. Oversampling is accomplished by zero padding: 7 zeros are inserted between any two succeding samples of the I and Q data stream. The excess bandwidth of the RRCF is chosen $r = 0.5$. The filtered data are applied to the inputs of two DACs. The unwanted high-frequency signals caused by digital-to-analog conversion are removed by two analog low-pass 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 done digitally are performed by digital circuits.

9.5.2 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. 9.10. With QPSK, we no longer have to deal with one single carrier, but 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_{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 ω_1 is the radian frequency of the carrier and θ_1 is the phase. The Costas loop was extended to lock onto a modulated QPSK signal.[1,20] This circuit is shown in Fig. 9.18. It is quite similar to the original arrangement (Fig. 9.10). The two low-pass filters (loop filters) in the I and Q branches are now followed by limiters. By virtue of the limiters the output signals become independent of the levels of the modulating signals, so the circuit can also be used to demodulate QAM signals (see Sec. 9.6). When the loop has locked, the VCO will generate a signal of the form

$$u_{\text{VCO}}(t) = \sin(\omega_1 + \theta_2)$$

where θ_2 is the phase of the output signal. Following the analysis of Gardner,[1] the output v_d of the adder (at center right in Fig. 9.18) is given by

$$\begin{aligned} v_d(t) = {} & [m_1(t)\sin\theta_e + m_2(t)\cos\theta_e]\,\text{sgn}[m_1(t)\cos\theta_e - m_2(t)\sin\theta_e] \\ & - [m_1(t)\cos\theta_e - m_2(t)\sin\theta_e]\,\text{sgn}[m_1(t)\sin\theta_e + m_2(t)\cos\theta_e] \end{aligned}$$

where 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

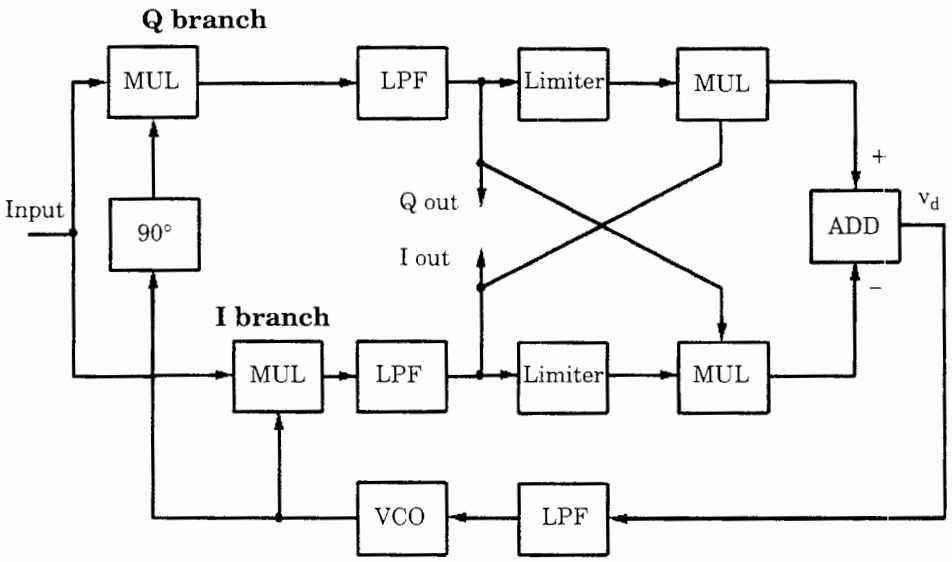


Figure 9.18 Costas loop extended for the demodulation of QPSK.

$$v_d(t) = \begin{cases} \sin\theta_e & \text{if } -45° < \theta_e < 45° \\ -\cos\theta_e & \text{if } 45° < \theta_e < 135° \\ -\sin\theta_e & \text{if } 135° < \theta_e < 225° \\ \cos\theta_e & \text{if } 225° < \theta_e < 315° \end{cases}$$

There are four values for θ_e where the loop can lock: $\theta_e = 0°$, 90°, 180°, and 270°, respectively. If it locks at $\theta_e = 0^0$, $v_d(t) = \sin\theta_e$, which can be approached by θ_e for small phase error. The Costas loop then operates like a conventional PLL. If the loop locks at $\theta_e = 90°$, $v_d(t)$ becomes $-\cos\theta_e$, which is near zero. When the phase error deviates only slightly from 90°, $v_d(t)$ is proportional to that deviation, and the circuit again operates like a simple PLL. Analog operations 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.[42] Figure 9.19 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 takes 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.

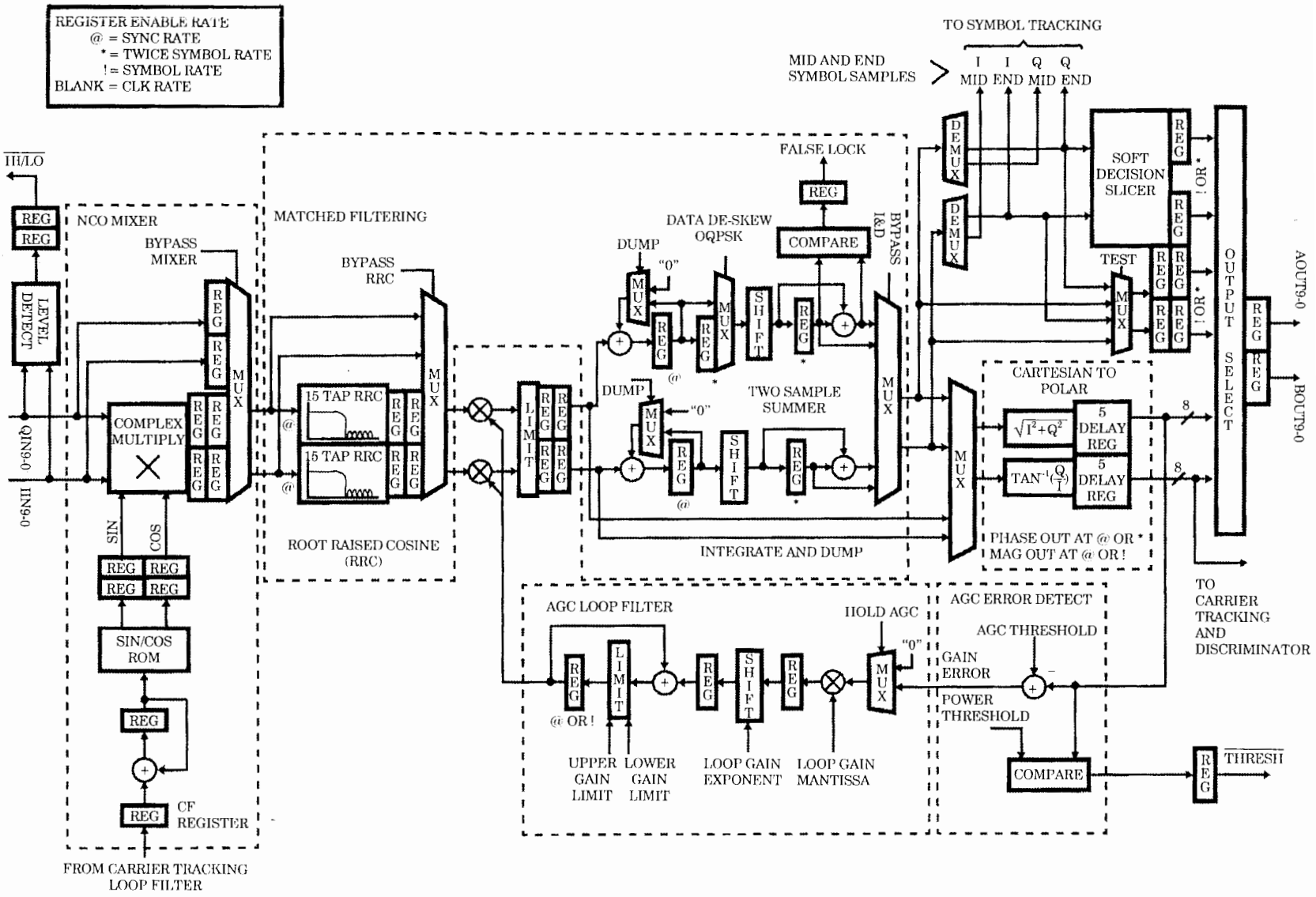


Figure 9.19 Block diagram of the digital Costas loop Harris HSP50210. *(Used with permission of the Harris Corporation.)*

The sine and cosine waveforms that are required for demodulation are taken from a table lookup ROM, shown at 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, which can be used when the data signals are rectangular. More detailed information is available from application notes.[45,46]

9.6 Digital Communications Using QAM

As was demonstrated in Sec. 9.2, QAM can be considered an extension to QPSK. While all symbols have the same amplitude with QPSK, the amplitude of both in-phase and quadrature components now can take on a number of different values with QAM. 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 increased bandwidth. A nice application of QAM is found in digital video broadcasting (DVB). In Europe, a proposal for digital video broadcasting was presented under the name DVB-C.[47] 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.[20] Source coding is a means to reduce the symbol rate at the data source. An analog TV signal has a bandwidth of approximately 5 MHz. When transmitting the signal digitally, we therefore would have to sample it at 10 MHz. Using 8 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 BPSK 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 Reed-Solomon codes.[20] 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. With QAM_{64}, 6 bits per symbol can be transmitted (refer to Fig. 9.4). Consequently, the symbol rate becomes (41.4 Mbit/s)/6 = 6.9 Mbaud. When transmitting the signal in the baseband, a one-sided bandwidth of 6.9/2 = 3.45 MHz would be required. Because the proposal recommends the RRCF (with $r = 0.15$) for filtering the data, the bandwidth is slightly increased to $(1 + r)3.45 = 3.97$ MHz. When a high-frequency carrier is modulated with that signal, its two-sided bandwidth becomes 2(3.97) = 7.94 MHz, which is below the required value of 8 MHz.

9.7 Digital Communications Using FSK

9.7.1 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 us recall what should be achieved theoretically, and why the simpler circuits cannot reach these goals.

As stated in Sec. 9.2, there are two basic types of FSK decoding—coherent and noncoherent. Equation (9.1) says 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 FSK decoder (as in Fig. 6.21, for example), it would have to lock onto a new frequency whenever an imcoming symbol differs from the preceding. 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. 6.21 sets the D flipflop on every positive edge of the input signal u_1 and hence needs at least one cycle to react onto a frequency step. To accomplish the required symbol rate, the frequencies f_1 and f_2 would have to be chosen much higher. This would result in 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; that is, 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_2 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, and hence can be written as

$$s_1(t) = (\cos 2\pi f_1 t)\,\text{rect}(t/T_S) \tag{9.9}$$

where $\text{rect}(t/T_S)$ denotes a rectangular pulse of duration T_S and is defined by

$$\text{rect}(t/T_S) = \begin{cases} 1 & \text{for } -T_S/2 \le t \le T_S/2 \\ 0 & \text{for } |t| > T_S/2 \end{cases}$$

The spectrum of tone 1 is a sinc function

$$S_1(f) = T_s \frac{\sin(\pi T_S(f - f_1))}{\pi T_S(f - f_1)} \tag{9.10}$$

The spectrum is symmmetrical about the frequency $f = f_1$, as shown in Fig. 9.20 (solid curve). It has zeros at frequencies $f = f_1 \pm k/T_s$, where k is an integer >0. 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. 9.20. 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 that 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 are going now to discuss some of the most familiar coherent and noncoherent FSK decoder schemes.

9.7.2 Coherent FSK detection

A coherent binary FSK decoder is shown in Fig. 9.21. 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 imple-

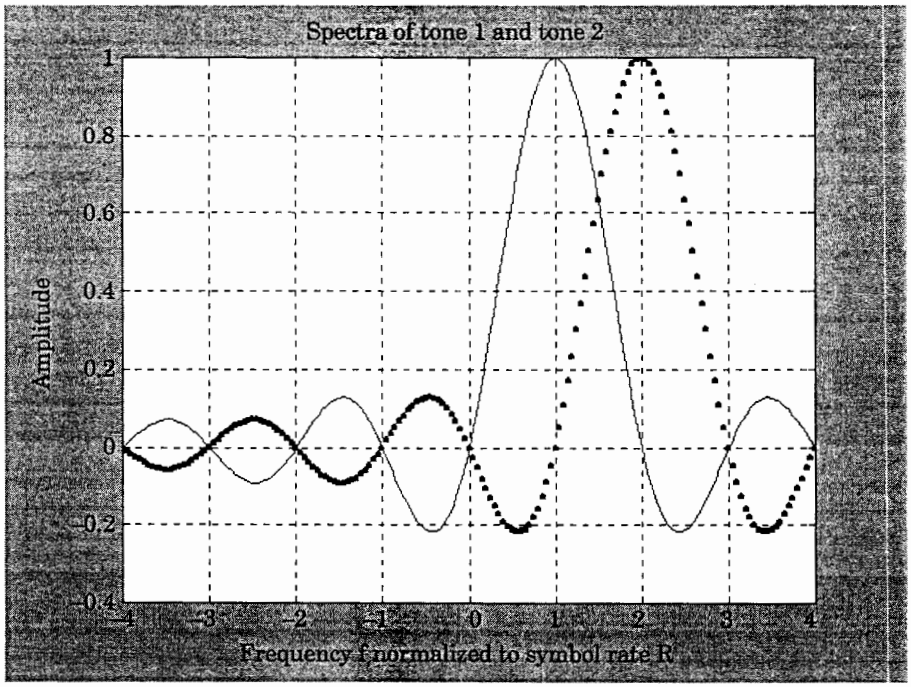


Figure 9.20 Spectra of tone 1 and tone 2 in noncoherent FSK detection.

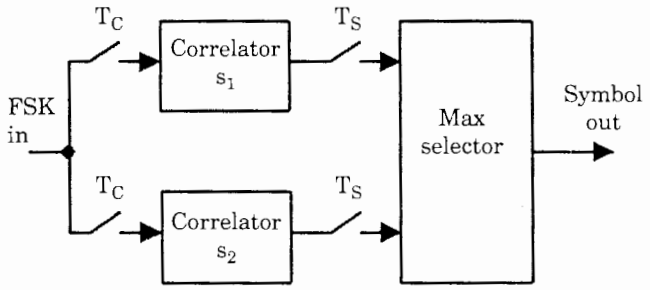


Figure 9.21 Coherent FSK decoder.

mented by FIR filters; their coefficients are identical to those of the samples of the corresponding symbol waveforms; see Fig. 9.5*b*. 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 will then output 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 m-ary FSK, more than two frequencies are used to represent m-ary

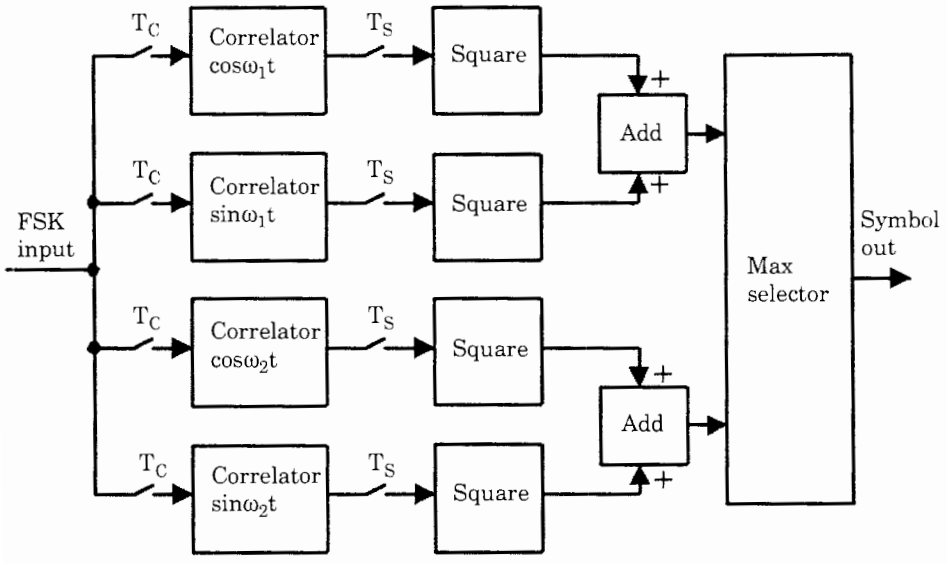


Figure 9.22 Noncoherent (quadrature) FSK decoder.

symbols, e.g., four. For arbitrary m, m correlation filters must be provided in the FSK decoder.

9.7.3 Noncoherent FSK detection and quadrature FSK decoders

A noncoherent binary FSK decoder is shown in Fig. 9.22. 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 ω_1 and ω_2. Consequently the decoder provides four waveforms; i.e., $\cos\omega_1 t$, $\sin\omega_1 t$, $\cos\omega_2 t$, and $\sin\omega_2 t$. It is still assumed that radian frequency ω_1 represents a logical 0, and radian frequency ω_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 φ is an arbitrary phase. If φ 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 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 φ 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 φ, 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 φ. 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 symbol end. If it happens that the upper correlator has a larger output, the decoder decides that the received symbol is 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.

Chapter

10

State of the Art of Commercial PLL Integrated Circuits

The designer of a PLL system has a broad choice of ICs built from different semiconductor technologies. The first available ICs were mainly manufactured in "bipolar" technology; i.e., they used a process known from bipolar operational amplifiers. Shortly after, transistor-transistor logic (TTL) circuits appeared, and somewhat later complementary metal-oxide-semiconductor (CMOS) devices became available. Faster circuits were produced in emitter-coupled logic (ECL) technology. In the late 1980s, some gallium arsenide (GaAs) PLLs were added to the already big family of silicon devices. The GaAs devices extended the frequency range to the gigahertz region. The GaAs technology could not realize the expected breakthrough, however; in recent years, silicon and SiGe devices came on the market that outperformed even the GaAs types in speed. In the last few years, production of GaAs devices has been discontinued.

There are a great number of ICs that contain a complete PLL system. In addition, there exist ICs that 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. Besides the usual PLL components, the synthesizer ICs mostly include fixed- or variable-ratio down-scalers for the reference input and one or more programmable down-scalers for the VCO output signal. The first of these down-scalers is used to scale down the frequency of a quartz oscillator that typically operates at 5 or 10 MHz. Most of these synthesizer ICs include the circuitry of the reference oscillator.

In the last few years, 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 very fast.

Many of the offered ICs contain some extras such as an additional op-amp, an additional in-lock detector, and the like. Table 10.1 lists the presently available ICs containing a full PLL, a full frequency synthesizer, or parts of a PLL

system. The PLL ICs are sorted by device family, which is listed in the first column. The second column describes the function of the IC. When looking for an ADPLL circuit, for example, we recognize that only one type is on the market, the 74xx297. It is available in two device technologies, however: in CMOS and in LS-TTL. Furthermore, the general-purpose DPLL chips are all based on the old 4046 chip, which is a member of the 4000 CMOS family originally introduced by RCA. The 4046 contains two phase detectors: an EXOR and a PFD. Three descendants have been derived from this circuit: the 74HC/HCT7046, the 74HC/HCT4046, and the 74HCT9046. The 74HC/HCT7046 is functionally equivalent to the former 4046, but the 74HC/HCT4046 is different, because it contains three different phase detectors instead of two: an EXOR, a JK-flipflop, and a PFD. The 74HCT9046 is similar to the 74HCT7046, but the PFD integrated in this chip provides charge pump outputs (see Sec. 2.7). In addition, a bandgap reference has been added which controls the frequency generated by the VCO. Using a stable source improves the stability of the VCO output frequency. Many of the newer frequency synthesizer PLLs include phase detectors with charge pump output.

The choice is largest among the LPLL chips. Besides the PLL systems, the market offers a wide selection of related circuits such as phase detectors, VCOs, prescalers, and frequency synthesizers.

Searching for PLL devices has become simple because most of them are listed in *IC Master*, a data book published yearly by Hearst Business Communications, Garden City, New York. This library is also available on CD-ROM and can be accessed from http://www.icmaster.com free of charge. The PLL circuits are listed in the category of *linear circuits*. There are links to all PLL IC manufacturers, which enables the user to download almost every data sheet and a great number of application notes. There is also a link to a Web site created by National Semiconductor that performs frequency synthesizer designs based on the company's synthesizer ICs (http://www/national.com/appinfo/wireless).[50] This design feature can be accessed by calling the *EasyPLL* option on this Web site.

TABLE 10.1 Commercially Available PLL ICs and Related Components, 2002

Device technology	Function	Device	Source	Features
HC	Phase-locked loop (DPLL)	*74HC4046*	Fairchild Intersil On Semi Philips Rochester TI	General-purpose DPLL containing three phase detectors: EXOR, JK-flipflop, and PFD. Frequency range to 19 MHz.
HCT	Phase-locked loop (DPLL)	*74HCT4046*	Intersil Philips TI	As 74HC4046.
HC	Phase-locked loop (DPLL)	*74HC7046*	Philips TI Intersil	General-purpose DPLL containing two phase detectors: EXOR and PFD.
HCT	Phase-locked loop (DPLL)	*74HCT7046*	Philips TI	As 74HC7046.
4xxx	Phase-locked loop (DPLL)	*4046*	Fairchild Intersil Lansdale On Semi R&E Intl ST Micro TI	General-purpose DPLL containing two phase detectors: EXOR and PFD.
HCT	Phase-locked loop (DPLL)	74HCT9046	Philips	General-purpose DPLL containing two phase detectors: EXOR and PFD. PFD has charge pump output. VCO powered by band gap reference for high stability of center frequency.
HC	ADPLL	74HC297	Philips	General-purpose ADPLL. Has two phase detectors: EXOR and JK-flipflop. $\div N$ divider is external.
HCT	ADPLL	74HCT297	Philips Intersil	As 74HC297.
LS-TTL	ADPLL	74LS297	TI	As 74HC297.
TTL	Phase-frequency detector	4344	Lansdale	
Linear	Clock recovery and data retiming PLL	AD802-155 AD802-45 AD802-52 AD805 CLC016	AD AD AD AD National	For cordless phones.
Linear	PLL prescaler	TB2109 TB31201	Toshiba Toshiba	200–400 MHz, for cordless phones.
Linear	Count extender	SP8790*	Mitel	$\div 4$, $\div 10/11$ becomes $\div 40/41$.
Linear	Count extender	SP8794*	Mitel	$\div 8$, $\div 10/11$ becomes $\div 80/81$.
Linear	Frequency synthesizer	RCK604	IntCirSys	Crystal controlled.

TABLE 10.1 *Continued*

Device technology	Function	Device	Source	Features
Linear	Frequency synthesizer	MC145106	Motorola	Chip contains reference oscillator, ÷2 reference divider, and ÷2 reference output, another reference divider (scaling factor 2^9 or 2^{10}), phase detector, and 9-bit programmable divide-by-N counter. Programming by parallel data input (9 + 1 bits).
		SM5133*	NPC	
		SM5142A	NPC	
		SM5165	NPC	
		SM5166	NPC	
		LV2105V	Sanyo	
		LV2151V	Sanyo	
		LV2153V	Sanyo	
Linear	Frequency synthesizer	LV2158V	Sanyo	
		U2781B	TEMIC	
		TH7010	Thesys	
Linear	Frequency synthesizer and down-converter	UPB1004GS	NEC	Dual, 3 V.
Linear	Frequency synthesizer, bidirectional	SP5510*	Mitel	1.3 GHz, for TV/VCR tuning systems.
		SP5512	Mitel	
		SP5611	Mitel	
Linear	Frequency synthesizer, bidirectional	SP5055	Mitel	2.6 GHz, for satellite TV.
		SP5655	Mitel	2.7 GHz, for TV tuning systems.
CMOS	Frequency synthesizer	TDD1742T	Philips	
Linear	Frequency synthesizer/ controller	LC7185-8750	Sanyo	For CB transceiver.
Linear	Frequency synthesizer, direct	SP2002	Mitel	Up to 400 MHz, square, triangular, or sine wave output.
Linear	Frequency synthesizer	TDA7326	ST Micro	For AM/FM radio.
Linear	Frequency synthesizer	LC72121*	Sanyo	For AM/FM tuners.
		LC72131*	Sanyo	
		LC72132*	Sanyo	
		LC72133*	Sanyo	
		LC72134M	Sanyo	
		LC72135M	Sanyo	
		LC72136*	Sanyo	
		LC72137*	Sanyo	
		LC72144M	Sanyo	
		LC72146*	Sanyo	
		LC72191	Sanyo	
Linear	Frequency synthesizer	NJU6104	NJR	For DTS.
Linear	Frequency synthesizer	LC72140	Sanyo	For FM/AM tuner.
Linear	Frequency synthesizer	BU2611*	ROHM	For tuners.
		BU2614*	ROHM	
		BU2615*	ROHM	

TABLE 10.1 *Continued*

Device technology	Function	Device	Source	Features
		BU2616F	ROHM	
		BU2618FV	ROHM	
Linear	Frequency synthesizer	M64895	Mitsubishi	For TV/VCR tuners, I^2C-bus.
Linear	Frequency synthesizer	M54937	Mitsubishi	For TV/VCR tuners, serial input.
		M54938	Mitsubishi	
		M54939	Mitsubishi	
		M56768	Mitsubishi	
		M56769	Mitsubishi	
		M56770	Mitsubishi	
		M56771	Mitsubishi	
		M56773	Mitsubishi	
		M56776	Mitsubishi	
		M64092	Mitsubishi	
		M64093	Mitsubishi	
		M64892	Mitsubishi	
		M64893	Mitsubishi	
		M64894	Mitsubishi	
Linear	Frequency synthesizer	M64897GP	Mitsubishi	For TV/VCR tuners, with dc-dc converter.
		M64898GP	Mitsubishi	
Linear	Frequency synthesizer	TRF2040	TI	Fractional-N/integer-N operation, triple channel.
Linear	Frequency synthesizer	SA7026	Philips	Fractional N, dual, 1.3 GHz.
		SA8016	Philips	Fractional N, dual, 2.5 GHz.
		SA8026	Philips	
		SA7015	Philips	Fractional N, 1 GHz.
		SA7025	Philips	
		SA7016	Philips	Fractional N, 1.3 GHz.
		SA8025*	Philips	Fractional N, 1.8 GHz.
		SA8015	Philips	Fractional N, 2 GHz.
Linear	Frequency synthesizer	NJ88C33	Mitel	I^2C-bus.
Linear	Frequency synthesizer	SP5658*	Mitel	Low phase noise, 2.7 GHz.
		SP5659	Mitel	
Linear	Frequency synthesizer	SM5160*	NPC	
Linear	Frequency synthesizer	MK1711-01	IntCirSys	20 to 200 MHz.
Linear	Frequency synthesizer	SY89429V	Micrel	31.25 to 510 MHz.
Linear	Frequency synthesizer	SY89430V	Micrel	50 to 950 MHz.
ECL	Frequency synthesizer	MC12181	Motorola	125 to 1000 MHz.
Linear	Frequency synthesizer	MB87094	Fujitsu	Serial input.
4xxx	Frequency synthesizer	MC145156-2	Motorola	Programmable reference divider (scaling factor 8/64/128/256/640/1000/1024/2048), programmable $\div N$ counter (3 to 1023) and programmable $\div A$ counter (0 to 127). Digital programming of A and N by serial input signal (10 + 7 bits). Uses external 2-modulus prescaler.

TABLE 10.1 *Continued*

Device technology	Function	Device	Source	Features
		MC145157-2	Motorola	Chip contains reference oscillator, 2 phase detectors, in-lock detector, 14-bit programmable reference divider, and 14-bit programmable divide-by-N counter. Programming by serial data input.
		MC145158-2	Motorola	Programmable reference divider (scaling factor 3 to 16384), oscillator, PFD, in-lock detector, 10-bit programmable $\div N$ counter, programmable $\div A$ counter, uses external 2-modulus prescaler. Programming by serial signal.
Linear		PLL0305* PLL2001 SM5158A	NPC NPC NPC	
4xxx	Frequency synthesizer	MC14159-1	Motorola	Serial input, with analog phase detector.
Linear	Frequency synthesizer	UMA1014	Philips	For mobile phones.
Linear	Frequency synthesizer	SP8861 SP8853* SP8858 SP8855D SP8855E	Mitel Mitel Mitel Mitel Mitel	1.3 GHz. 1.3/1.5 GHz. 1.5 GHz. 1.7 GHz. 2.7 GHz.
Linear	Frequency synthesizer	MB87093A MB87095A MB87096A	Fujitsu Fujitsu Fujitsu	With power-saving function.
4xxx	Frequency synthesizer	MC145170 MC145170-1 MC145170-2	Motorola Motorola Motorola	With serial interface.
Linear	Frequency synthesizer	TBB206	Infineon	With 3-line bus.
Linear	Frequency synthesizer	UMA1005T ST7162	Philips ST Micro	Dual.
4xxx	Frequency synthesizer	MC145162 MC145162-1 MC145165 MC145166 MC145167 MC145168 MC145169	Motorola Motorola Motorola Motorola Motorola Motorola Motorola	Dual, for cordless phones, 60 MHz.
Linear	Frequency synthesizer	BU2630*	ROHM	Dual PLL.
Linear	Frequency synthesizer	UMA1021* UMA1022M UMA1020M	Philips Philips Philips	Dual, up to 2.2 GHz. Dual, up to 2.4 GHz.
4xxx	Frequency synthesizer	MC145225 MC145230	Motorola Motorola	Dual, with DACs and voltage multipliers, 2 GHz/280 MHz.

TABLE 10.1 *Continued*

Device technology	Function	Device	Source	Features
4xxx	Frequency synthesizer	MC145149	Motorola	Dual, with serial interface.
Linear	Frequency synthesizer	WB1315	IC Works	Dual, with 2.5-GHz prescalers.
Linear	Frequency synthesizer	M64074	Mitsubishi	Dual 1 GHz.
Linear	Frequency synthesizer	LMX2335*	National	Dual, 1.1 GHz/1.1 GHz.
ECL	Frequency synthesizer	MC12302	Motorola	Dual, 1.1 GHz/500 MHz.
Linear	Frequency synthesizer	PE3282	Peregrine	Dual, 1.1 GHz/510 MHz.
		LMX2332*	National	Dual, 1.2 GHz/510 MHz.
		PE3292	Peregrine	Dual, 1.2 GHz/550 MHz.
		UMA1018M	Philips	Dual, 1.25 GHz.
		LMX2336*	National	Dual, 2 GHz/1.1 GHz.
		LMX2331*	National	Dual, 2 GHz/510 MHz.
ECL	Frequency synthesizer	MC12306	Motorola	Dual, 2.0 GHz/500 MHz.
		MC12310	Motorola	Dual, 2.5 GHz/500 MHz.
Linear	Frequency synthesizer	LMX2330*	National	Dual, 2.5 GHz/510 MHz.
		HFA3524	Intersil	Dual, 2.5 GHz/600 MHz.
		S4503	AMCC	Dual, 10 to 300 MHz.
		UMA1015*	Philips	Dual, 50 to 1100 MHz.
		NJ88C50	Mitel	Dual, 125 MHz (AMPS, ETACS, DECT).
		M54958	Mitsubishi	Dual, 400 MHz.
		LMX2337	National	Dual, 550 MHz/550 MHz.
		ICS9502	IntCirSys	Dual, 900 MHz/500 MHz.
		TH7023	Thesys	Dual, 1200 MHz/520 MHz, with prescaler.
4xxx	Frequency synthesizer	MC145173	Motorola	Dual band with ADC/frequency counter
Linear	Frequency synthesizer	SY89424V	Micrel	1 GHz.
		SP5026	Mitel	1 GHz, for TV/VCR tuning systems.
4xxx	Frequency synthesizer	MC145192	Motorola	1.1 GHz.
Linear	Frequency synthesizer	LMX1501A	National	1.1 GHz.
		LMX9301	National	1.1 GHz.
4xxx	Frequency synthesizer	MC145190	Motorola	1.1 GHz, with PFD and prescaler.
		MC145191	Motorola	1.1 GHz, with 5-V phase detector.
Linear	Frequency synthesizer	TSA5511	Philips	1.3 GHz, for TV tuning systems.
		TSA5512	Philips	
		TSA5514	Philips	
		TSA5515T	Philips	
	Frequency synthesizer	SP5024	Mitel	1.3 GHz, for TV/VCR tuning systems.
		TSA5520	Philips	
		TSA5521	Philips	
		TSA5526*	Philips	
		TSA5527*	Philips	
	Frequency synthesizer	TSA5522	Philips	1.4 GHz, for TV/VCR tuning systems.
	Frequency synthesizer	TSA5523M	Philips	1.4 GHz, I^2C-bus.
	Frequency synthesizer	SP8852D	Mitel	1.7 GHz.
		SP8854D	Mitel	

TABLE 10.1 *Continued*

Device technology	Function	Device	Source	Features
	Frequency synthesizer	Q3236	Qualcomm	2 GHz.
		CXA1787	Sony	2 GHz, for mobile phones.
4xxx	Frequency synthesizer	MC145200	Motorola	2 GHz with 8- to 9.5-V phase detector.
Linear	Frequency synthesizer	SP5070	Mitel	2.4 GHz, for satellite receivers.
		SP5054	Mitel	2.6 GHz, for satellite TV tuning systems.
	Frequency synthesizer	SP8852E	Mitel	2.7 GHz.
		SP8854E	Mitel	
		M64896	Mitsubishi	
	Frequency synthesizer	SP5657	Mitel	2.7 GHz, for satellite TV.
		SP5668	Mitel	2.7 GHz, for TV/VCR tuning systems.
		SP5654	Mitel	2.7 GHz, 3-wire bus.
		SP5502	Mitel	1.3 GHz, 4-bit address.
		SP5511*	Mitel	1.3 GHz, 4-bit address, for TV/VCR tuning systems.
4xxx	Frequency synthesizer	MC145145-2	Motorola	4-bit data bus input. Chip contains reference oscillator, 12-bit programmable reference divider, 2 phase detectors, in-lock detector, 14-bit programmable divide-by-N counter, and data latches for parameter storage.
		MC145146-2		4-bit data bus input. Programmable reference divider (÷3 to 4096), oscillator, PFD, in-lock detector, programmable ÷N counter, programmable ÷A counter (for external 2-modulus prescaler).
Linear	Frequency synthesizer	PCK604	OnChipSys	50 to 600 MHz.
		SC11346	Int. Semi	80 MHz.
		AD809	AD	155.52 MHz.
		LMX2301	National	160 MHz, for RF communications.
		SY89429A	Micrel	400 to 800 MHz.
		M56760	Mitsubishi	500 MHz.
		LMX2305	National	550 MHz, for RF communications.
		M64080	Mitsubishi	410 MHz.
		LC72130	Sanyo	For FM/AM tuner.
Linear	Intercarrier modulator	4002Y-A	NCM	Internal PLL and comparator.
ECL	Mixer	MC12002	Motorola	Double balanced, analog mixer.
4xxx	Phase detector	IMI4345	IMI	
ECL	Phase-frequency detector	12540	Lansdale	
		MCH12140	Motorola	
		MCK12140	Motorola	
		MC12040	Motorola	
Linear	PLL with prescaler	MB1501H	Fujitsu	1.1-GHz prescaler, 2-modulus, ÷64/65 or ÷128/129.
	PIF processor/PLL	μPC1820	NEC	

TABLE 10.1 *Continued*

Device technology	Function	Device	Source	Features
Linear	PLL	NE565 SE565	Philips	Frequency range 0 to 500 kHz. VCO has parallel square and triangular wave outputs.
		XR215*	Exar	
		XR2212*	Exar	
		SL652	Mitel	LPLL having 4 timing current inputs and 2 binary control inputs for switching of timing currents. Main application is FM, FSK, PSK modulator/demodulator, etc. Frequency range 0–500 kHz.
		MN6152	Panasonic	
		MN6153	Panasonic	
		MN6155	Panasonic	
		NE564	Philips	Postdetection processor and limiting circuit at the input of the PD integrated on chip. Frequency range to 50 MHz.
		SE564	Philips	
		LC7150	Sanyo	
		LC7152	Sanyo	
Linear	PLL and prescaler	MB1503	Fujitsu	1 GHz.
Linear	PLL-based zero delay buffer	MPC961*	Motorola	
Linear	PLL clock driver	MC88LV930	Motorola	
		MC88LV950	Motorola	
		MC88LV970	Motorola	
		CDC2509*	TI	
		CDC2510*	TI	
		CDC2516	TI	
		CDC509	TI	
		CDC516	TI	
Linear	PLL clock driver	MPC932	Motorola	Programmable.
		MC88PL117	Motorola	Power PC 601, Pentium applications.
		SN54CDC586	TI	3 state outputs.
	PLL clock recovery and data retiming	AD800	AD	
		AD800-45	AD	45 MHz.
		AD800-52	AD	52 MHz.
		AD800-155	AD	155 MHz.
		AD803	AD	30 Mbps.
Linear	PLL controller for DC motor drive	UC1637	Unitrode	
		UC2637	Unitrode	
		UC3637	Unitrode	
Linear	PLL demodulator	LMC568	National	FM/FSK demodulator.
CMOS	PLL for radio receiver	LC7218	Sanyo	
NMOS	PLL for radio receiver	LM7001	Sanyo	

TABLE 10.1 *Continued*

Device technology	Function	Device	Source	Features
Linear	PLL frequency controller	UC1633	Unitrode	PLL for motor speed control. Contains sense amplifier for speed feedback signal (from Hall sensor or other speed detector).
		UC1634	Unitrode	Similar to UC1633, but is intended to drive two-phase brushless motors.
		UC1635	Unitrode	
		UC2633	Unitrode	
		UC2634	Unitrode	
		UC2635	Unitrode	
		UC3633	Unitrode	
		UC3634	Unitrode	
		UC3635	Unitrode	
Linear	Frequency synthesizer	MB1503/13	Fujitsu	
		HPLL8001	HP	
		PMB2307R	Infineon	
		TBB200	Infineon	
		SM5162	NPC	
		SM5168	NPC	
		SM5170A	NPC	
Linear	Frequency synthesizer, clock chip	IMISC434	IMI	
		IMISC464	IMI	
		IMISC465	IMI	
		IMISC466	IMI	
		IMISC468	IMI	
		IMISC470	IMI	
		IMISC471	IMI	
		IMI2XFSUT157	IMI	Two independent serial input PLL frequency synthesizers equivalent to two IMI145157's on one chip. Input frequency >60 MHz (typical).
Linear	Frequency synthesizer	MK1574-01*	IntCirSys	For frame rate communications.
		TDA8735*	Philips	For satellite receivers.
		MK2049-01	IntCirSys	Generates T1, E1, T3, E3, and OC3 frequencies.
		SM5160	NPC	High speed.
		ILC1080	Impala Linear	High speed, 40 to 120 MHz.
		MB15C101	Fujitsu	IF band.
		MB15C103	Fujitsu	
		IMI4347	IMI	Phase detector.
		PMB2306	Infineon	Serial control.
ECL	Frequency synthesizer	MC12179	Motorola	Serial input.
		MC12202	Motorola	
		MC12206	Motorola	
		MC12210	Motorola	
		MC122179	Motorola	
Linear	Frequency synthesizer	SM5158	NPC	Serial input, high speed.
		GM6530	Hyundai	Serial interface.
		MB15E03SL	Fujitsu	With 1.2-GHz prescaler.

TABLE 10.1 *Continued*

Device technology	Function	Device	Source	Features
		MB15E05SL	Fujitsu	Single serial input, 2.0 GHz.
		MB15E07SL	Fujitsu	
		MB15F02SL	Fujitsu	Dual serial input.
		MB15F03SL	Fujitsu	Dual serial input.
		MB15F07SL	Fujitsu	Dual serial input.
		MB15F08SL	Fujitsu	Dual serial input.
		IMI145145	IMI	4-bit data bus input. Chip contains reference oscillator, 12-bit programmable reference divider, 2 phase detectors, in-lock detector, 14-bit programmable divide-by-N counter, and data latches for parameter storage.
Linear	PLL	SM5153	NPC	Fixed-ratio frequency dividing.
		SM162LF1S	NPC	
	PLL + prescaler	MB15F02*	Fujitsu	Dual, 1.2 GHz, ÷64/65 or ÷128/129.
		MB15F03L	Fujitsu	Dual, 1.8 GHz, ÷64/65 or ÷128/129.
		MB15F05L	Fujitsu	
		MB15F03	Fujitsu	Dual, 2 GHz, ÷64/65 or ÷128/129.
		MB15F04	Fujitsu	
		MB1504*	Fujitsu	0.5 GHz.
		MB1505	Fujitsu	
		MB1507	Fujitsu	
		MB1508	Fujitsu	
		MB1509	Fujitsu	
		MB1519	Fujitsu	
		MB15A01	Fujitsu	1 GHz.
		MB15A02	Fujitsu	
		MB15A16	Fujitsu	
		MB15B01	Fujitsu	
		MB15B03	Fujitsu	
		MB15B11	Fujitsu	
		MB15B13	Fujitsu	
		MB1501	Fujitsu	
		MB1502	Fujitsu	
		MB1502H	Fujitsu	
		MB1510	Fujitsu	
		MB1511	Fujitsu	
		MB1512	Fujitsu	
		MB1513	Fujitsu	
		MB1516A	Fujitsu	
		WB1332	IC Works	1.2 GHz.
		MB15E03*	Fujitsu	1.2 GHz, ÷64/65 or ÷128/129.
		MB15A19	Fujitsu	1.5 GHz.
		MB1514	Fujitsu	
		MB1506	Fujitsu	2 GHz.
		MB1517*	Fujitsu	
		MB15E05*	Fujitsu	2 GHz, ÷64/65 or ÷128/129.
		MB15E06	Fujitsu	2.5 GHz.
		MB1515	Fujitsu	
		MB1518	Fujitsu	

TABLE 10.1 *Continued*

Device technology	Function	Device	Source	Features
		MB15E07*	Fujitsu	2.5 GHz, ÷32/33 or ÷64/65.
		MB15C03	Fujitsu	120 MHz, ÷64/65.
		MB15S03	Fujitsu	178 MHz, ÷16/17.
		MB15S01	Fujitsu	259 MHz, ÷16/17.
		MB15S02	Fujitsu	284 MHz, ÷16/17.
	PLL programmable divider	MB87086A	Fujitsu	Binary 10-bit counter.
		MB87087	Fujitsu	Binary 14-bit counter.
NMOS	PLL	LM7005	Sanyo	Ranges to UHF.
Linear	Frequency synthesizer	IMI145155	IMI	Programmable reference divider (scaling factor 16/512/1024/2048/3668/4096/6144/8192), ÷N counter with scaling factor of 3 to 16383. Scaling factor N digitally controlled by serial input signal (14 bits).
		IMI145157	IMI	Chip contains reference oscillator, 2 phase detectors, in-lock detector, 14-bit programmable reference divider, and 14-bit programmable divide-by-N counter. Programming by serial data input.
		SAA1057	Philips	
4xxx	Frequency synthesizer	IMI145106A	IMI	Parallel input.
		IMI145151	IMI	Parallel input (14 bits). Programmable reference divider (÷8/128/256/512/1024/2048/2410/8192) and ÷N divider with scaling factor 3 to 16383.
		MC145151-2	Motorola	Parallel input (14 bits). Programmable reference divider (÷8/128/256/512/1024/2048/2410/8192) and ÷N divider with scaling factor 3 to 16383.
		MC145152-2	Motorola	Programmable reference divider (scaling factors 8/64/128/256/512/1024/1160/2048), oscillator, PFD, in-lock detector, 10-bit programmable ÷N counter, 6-bit programmable ÷A counter for external 2-modulus prescaler. Fully parallel programming.
Linear		HC0320	Hughes	Programmable divider, to 1021 channels, adder, phase comparator.
		IMI145152	IMI	Programmable reference divider (scaling factors 64/128/256/512/1024/1160/2048), oscillator, PFD, in-lock detector, 10-bit programmable ÷N counter, 6-bit programmable ÷A counter for external 2-modulus prescaler. Fully parallel programming.

TABLE 10.1 *Continued*

Device technology	Function	Device	Source	Features
		IMI145156*	IMI	Programmable reference divider (scaling factor 8/64/128/256/640/ 1000/1024/2048), programmable ÷N counter (3 to 1023) and programmable ÷A counter (0 to 127). Digital programming of A and N by serial input signal (10 – 7 bits). Uses external 2-modulus prescaler.
		IMI145158*	IMI	Programmable reference divider (scaling factor 3 to 16384), oscillator, PFD, in-lock detector, 10-bit programmable ÷N counter, programmable ÷A counter, uses external 2-modulus prescaler. Programming by serial data input.
		MC145155-2	Motorola	Programmable reference divider (scaling factor 16/512/1024/2048/ 3668/4096/6144/ 8192), ÷N counter with scaling factor of 3 to 16383. Scaling factor N digitally controlled by serial input signal (14 bits).
		MB87001A	Fujitsu	Serial input, 13 MHz, 5 V, 3 mA.
		IMIFSUT157*	IMI	Serial input, 2 channel.
		IMI145146*	IMI	Programmable reference divider (÷3 to 4096), oscillator, PFD, in-lock detector, programmable ÷N counter, programmable ÷A counter (for external 2-modulus prescaler), 4-bit data input.
	PLL tuning circuit	MC44818	Motorola	1.3 GHz, for TV/VCR.
		MC44824	Motorola	
		MC44825	Motorola	
		MC44826	Motorola	
		MC44827*	Motorola	
		MC44828	Motorola	
		MC44829	Motorola	
		MC44871	Motorola	
		MC44802	Motorola	1.3-GHz prescaler.
		MC44864	Motorola	1.3-GHz prescaler, D/A converters.
	PLL, universal	KS8805B	Samsung	
	PLL, video dot clock generator	CH9073	Chrontel	
		AV9173	IntCirSys	
		AV9173-01	IntCirSys	
4xxx	PLL + prescaler	MC145220	Motorola	1.1 GHz, dual, ÷64/65 or ÷32/33.
Linear	PLL + prescaler	TEA8805	ST Micro	1.3 GHz, ÷8.
4xxx	PLL + prescaler	MC145201	Motorola	2 GHz, ÷64/65.
		MC145202	Motorola	
Linear		MB1501L	Fujitsu	1.1 GHz, ÷64/65 or ÷128/129.
	PLL, dual	SM1532*	NPC	60 MHz, for cordless phones.
		SM1534*	NPC	

TABLE 10.1 *Continued*

Device technology	Function	Device	Source	Features
	PLL, dual, with prescalers	WB1310	IC Works	1.2-GHz prescalers, serial input.
		WB1336	IC Works	2-GHz/1.1-GHz prescalers, serial input.
		WB1331	IC Works	2-GHz/510-MHz prescalers, serial input.
		WB1330	IC Works	2.7-GHz/510-MHz prescalers, serial input.
		WB1305	IC Works	510-MHz prescalers, serial input.
		WB1333	IC Works	
	PLL, dual	MB15U20	Fujitsu	1.1 GHz.
		U2782	Temic	
		SY89420	Micrel	1.12 GHz.
		SY89423	Micrel	
		U2783B	Temic	1250 MHz, 400 MHz.
		U2784B	Temic	2200 MHz, 400 MHz.
	PLL + prescaler	MB87014A	Fujitsu	÷64/65 or ÷128/129.
	PLL	SY89421V	Micrel	1.12 GHz.
	PLL + prescaler	WB1220	IC Works	2 GHz, serial input.
		WB1215	IC Works	1.2 GHz, serial input.
	PLL, dual, with prescalers	WB1337	IC Works	550 MHz.
	PLL + prescaler	WB1225	IC Works	2.5 GHz, serial input.
	PWM controller	MB3785	Fujitsu	
	Frequency synthesizer	MB87091	Fujitsu	Serial input, power saving function.
	PLL system block	MB87006A	Fujitsu	Serial input, 10 MHz, 3 to 5 V, 3.5 mA.
		MB87076	Fujitsu	Serial input, 15 MHz, 3 to 5 V, 3 mA.
		MB87086	Fujitsu	Serial input, 95 MHz, 3 to 5 V, 10 mA.
	Subcarrier PLL	MC44144	Motorola	For video applications.
	Frequency synthesizer	NJ8811	Mitel	For mobile radio, 2-device set.
		NJ8812	Mitel	
		SP8901	Mitel	
		SP8906	Mitel	
	Tone decoder	NE567	Philips	Operation to 500 kHz. Has additional quadrature phase detector that can be used as in-lock detector.
		KA567C	Samsung	
	VCO	CEM3340	OnChipSys	
		CEM3345	OnChipSys	
		AN8585	Panasonic	
		AN8586	Panasonic	
		AN8587	Panasonic	
		LC7444	Sanyo	
LS-TTL	VCO	SN74LS624	TI	
		SN74LS628	TI	
Linear	VCO + phase detector	LM565C	National	
	VCO buffer amplifier	KGF1145	OKI	
		KGF1146	OKI	
		RF2501	RF Micro	For ISM cellular.
		RF2502	RF Micro	
		KGF1191	OKI	300 MHz to 2.4 GHz.

TABLE 10.1 *Continued*

Device technology	Function	Device	Source	Features
	VCO function generator	XR2209*	Exar	
		LM566*	National	Frequency range 0–1 MHz. Simultaneous square and triangular wave output, no sine.
		XR2207*	Raytheon	Frequency range 0–1 MHz.
		XR2207*	Exar	Simultaneous square and triangular
		RC2207	Fairchild	wave output, no sine. Four timing resistors can be switched in and out by two binary inputs.
ECL	VCO, low power	MC12148	Motorola	
		MC12149	Motorola	
Linear	PLL	SL650	Mitel	Multiplier phase detector. Chip contains additional op-amp. Frequency range 0–500 kHz
		SL651	Mitel	As SL650, without op amp.
	VCO waveform generator	XR2206*	Exar	
		XR8038A	Exar	
		ICL8038	Intersil	Sine, square, triangular, sawtooth, and pulse output.
	VCO waveform generator, dual	CEM3371	OnChipSys	
	VCO, dual	CEM3374	OnChipSys	
LS-TTL		SN74LS625	TI	
		SN74LS629	TI	
		SN74LS124	TI	
Linear	Video dot clock generator	ICS1394	IntCirSys	
	Video PLL system	LM1291	National	For continuous-sync monitors.
		LM1292	National	
ECL	Voltage-controlled multivibrator	MC12101	Motorola	
		MC12100	Motorola	
	Voltage-controlled oscillator buffer	MC12147	Motorola	Low power.
Linear	2-modulus prescaler	SP8643	Mitel	÷10/11.

Note: Asterisks (*) preceding and/or following a part number mean that this device is registered with different prefixes and/or suffixes. Example: *4046* is delivered as CD4046B, MC14046B, CD4046BC, etc.

Chapter

11

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/π 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 4046 and the 7046 ICs (refer to Table 10.1), 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, ω_n, ζ, and many others are easily measured with the standard equipment available even in a hobbyist's lab, i.e., 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 device under test (DUT).

11.1 Measurement of Center Frequency f_0

Only the VCO portion of the DUT is used for this measurement (Fig. 11.1). 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 mea-

sured with an oscilloscope (Fig. 11.2*a*). If the values C_{ext} = 82 nF and R_0 = 2.2 kΩ are chosen for the external components, the VCO oscillates at a center frequency f_0 of 6.54 kHz.

11.2 Measurement of VCO Gain K_0

The same test circuit (Fig. 11.1) 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 Δu_f = 1 V. As shown in Fig. 11.2*b*, the VCO frequency falls to 5.78 kHz for Δu_f = 1 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

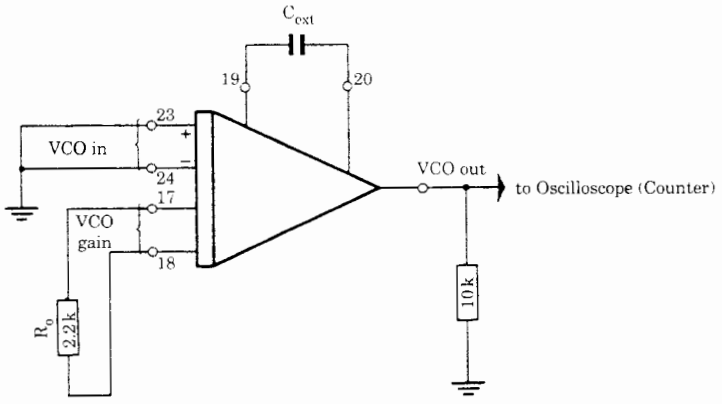


Figure 11.1 Test circuit for the measurement of the center frequency ω_0 and the VCO gain K_0.

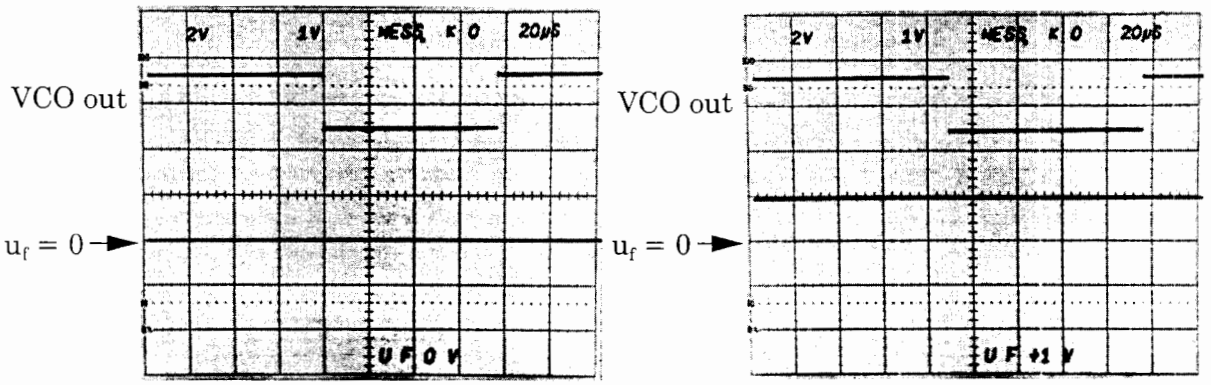


Figure 11.2 (*a*) Measurement of ω_0, u_f = 0 V; (*b*) measurement of K_0, u_f = 1 V.

$$K_0 = \frac{\Delta\omega_0}{\Delta u_f} = \frac{2\pi \cdot 0.76 \cdot 10^3}{1} = 4.78 \cdot 10^3 \text{ rad s}^{-1} \text{ V}^{-1}$$

The test shows furthermore that K_0 would be larger by a factor of 10 if ω_0 had been chosen larger by a factor of 10. Hence, for this device, K_0 varies in proportion to ω_0. When $R_0 = 2.2\,\text{k}\Omega$ is chosen, K_0 is given by the general equation

$$K_0 = 0.73 f_0 \text{ rad s}^{-1} \text{ V}^{-1} \tag{11.1}$$

where f_0 is in hertz. For example, for $f_0 = 1\,\text{kHz}$, we find $K_0 = 730$ rad $\text{s}^{-1}\text{V}^{-1}$.

11.3 Measurement of Phase Detector Gain K_d

The test circuit of Fig. 11.3 is used for the measurement of K_d. The PD of the XR-S200 is realized in 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. 11.4. In Fig. 11.4*a* the signal applied to phase detector input 1 (PD in #1) is a sine 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

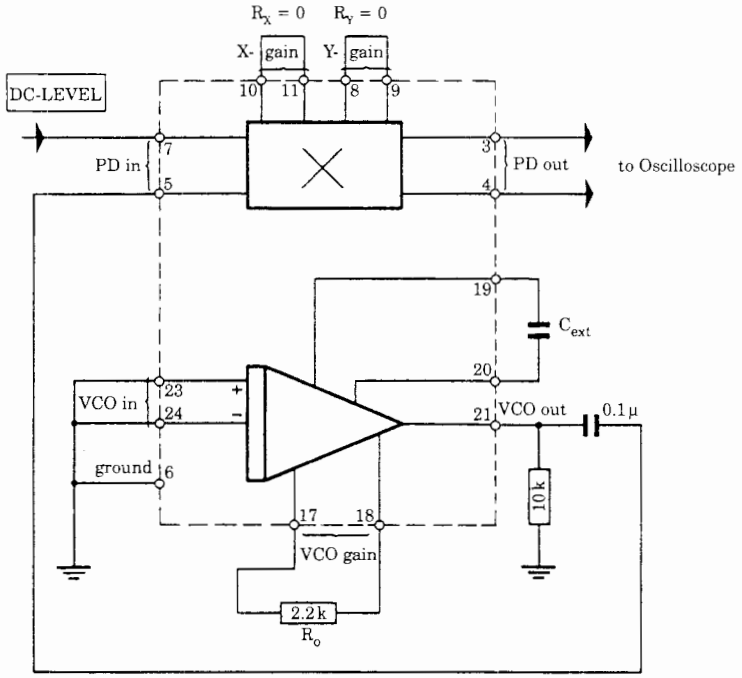


Figure 11.3 Test circuit for the measurement of phase detector gain K_d .

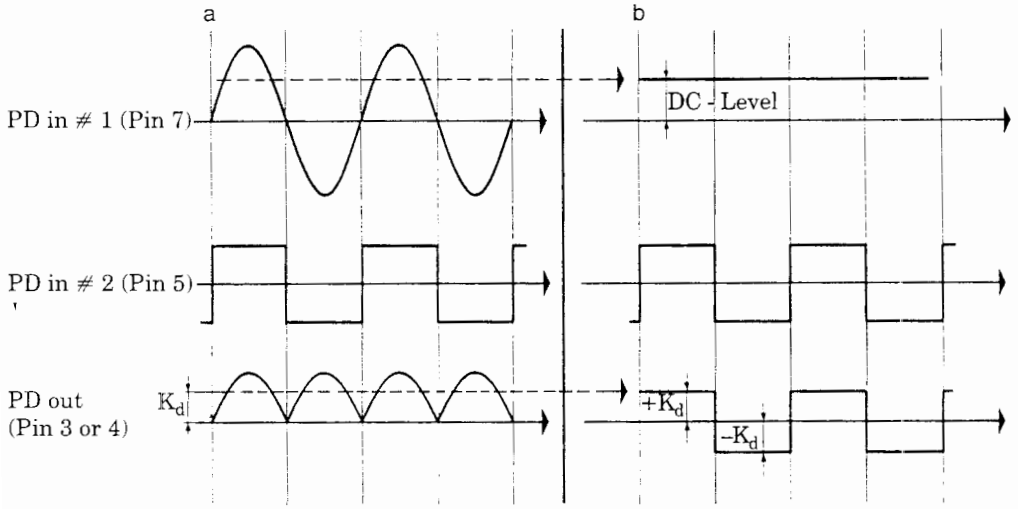


Figure 11.4 Waveforms of the phase detector output signal u_d for different signals applied to input 1. (*a*) For a sine-wave signal. (*b*) For a dc level.

both signals are in phase. Consequently the phase error θ_e is 90°. The phase detector output signal (PD out) is a full-wave rectified sine signal; its average value is given by

$$\overline{u_d} = K_d \sin 90° = K_d$$

i.e., 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. 11.4*b* 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 have to 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. 11.3 was measured at four different levels of the reference signal: at 10, 30, 40, and 100 mV rms. The measured signals are displayed in Fig. 11.5. The four oscillograms show the results for u_1 = 10, 20, 30, and 40 mV rms, respectively. As Fig. 11.3 demonstrates, this PD has a symmetrical output. Three signals are displayed in each oscillogram: 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).

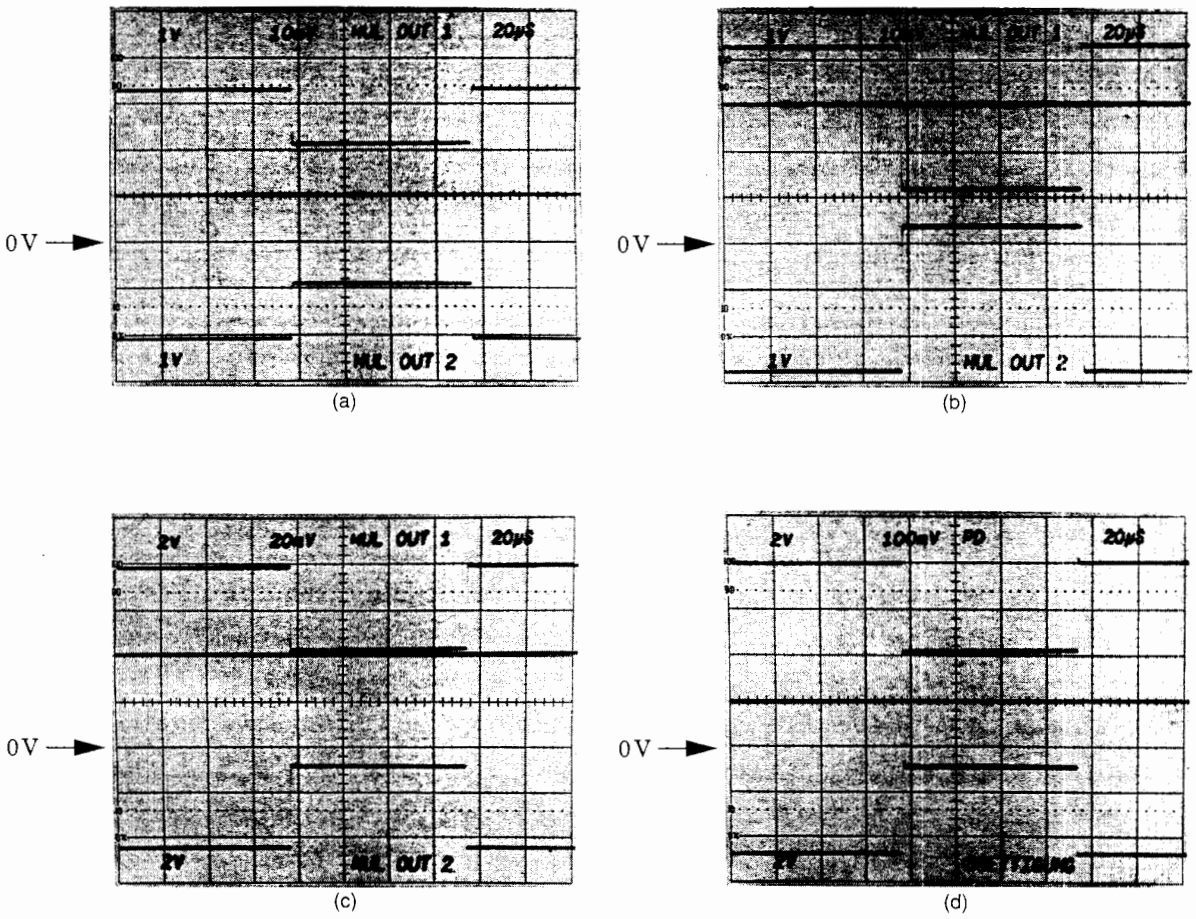


Figure 11.5 Waveforms measured with the test circuit of Fig. 11.3 for different dc levels applied to input 1. (*a*) 10 mV. (*b*) 30 mV. (*c*) 40 mV. (*d*) 100 mV.

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. 11.4) 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. 11.4.

The measured K_d values are plotted against the reference signal level in Fig. 11.6. It is clearly observed that the PD becomes saturated at signal levels above 400 mV rms.

11.4 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. 11.3 and by closing the

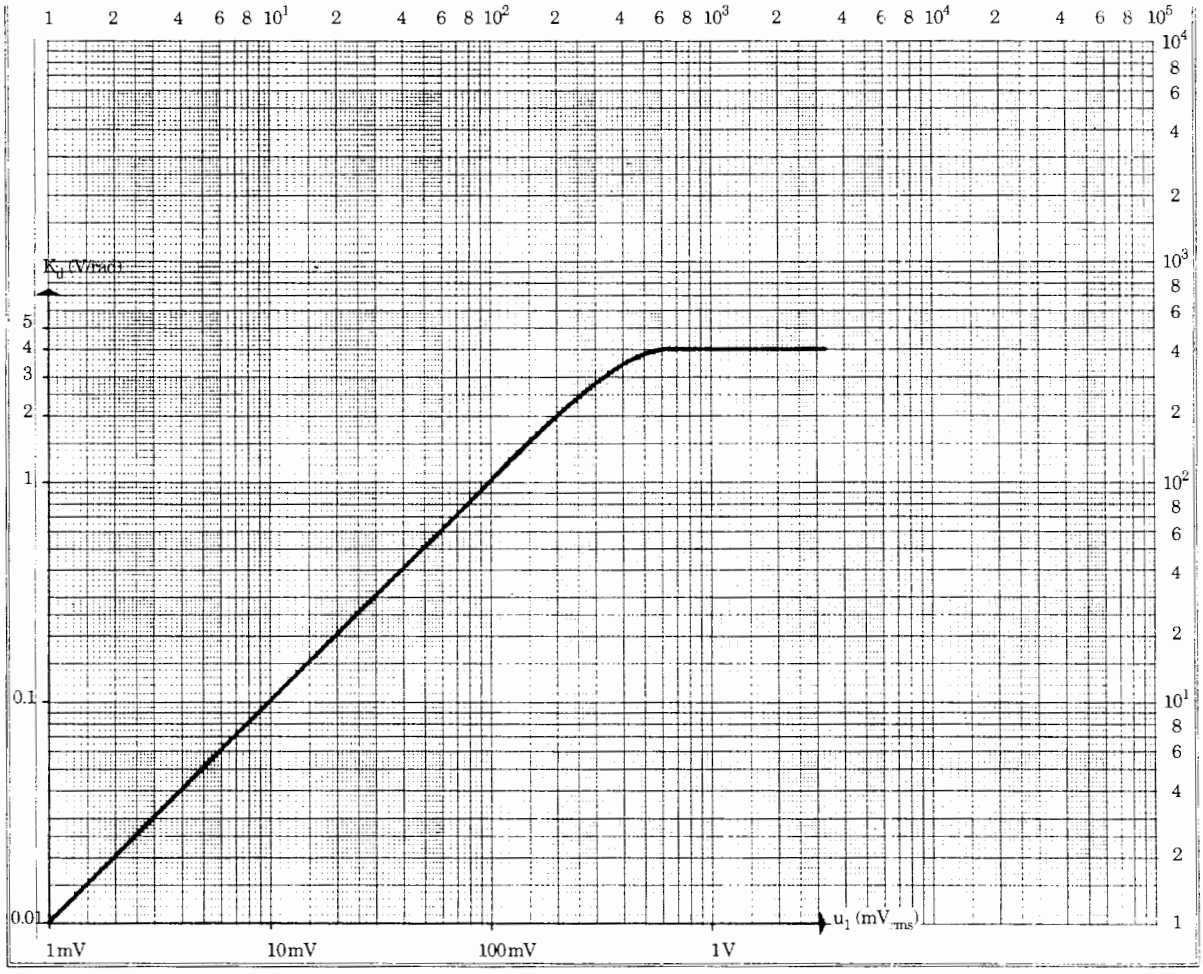


Figure 11.6 Plot of K_d against input voltage level in millivolts rms.

loop. The new test circuit of Fig. 11.7 is then obtained. This PLL does not use a down-scaler; hence we have $N = 1$ (N = scaling factor). A passive loop filter (Fig. 2.16) is chosen for simplicity; it consists of two on-chip resistors, $R_1 = 6\,\text{k}\Omega$ each, and the external capacitor C. (Note that resistor R_2 is set 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. 11.8). In the case of Fig. 11.8*a* the reference frequency f_1 is far away from the center frequency f_0 and the system is unlocked. Figure 11.8*b* 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 (Fig. 11.8*c*).

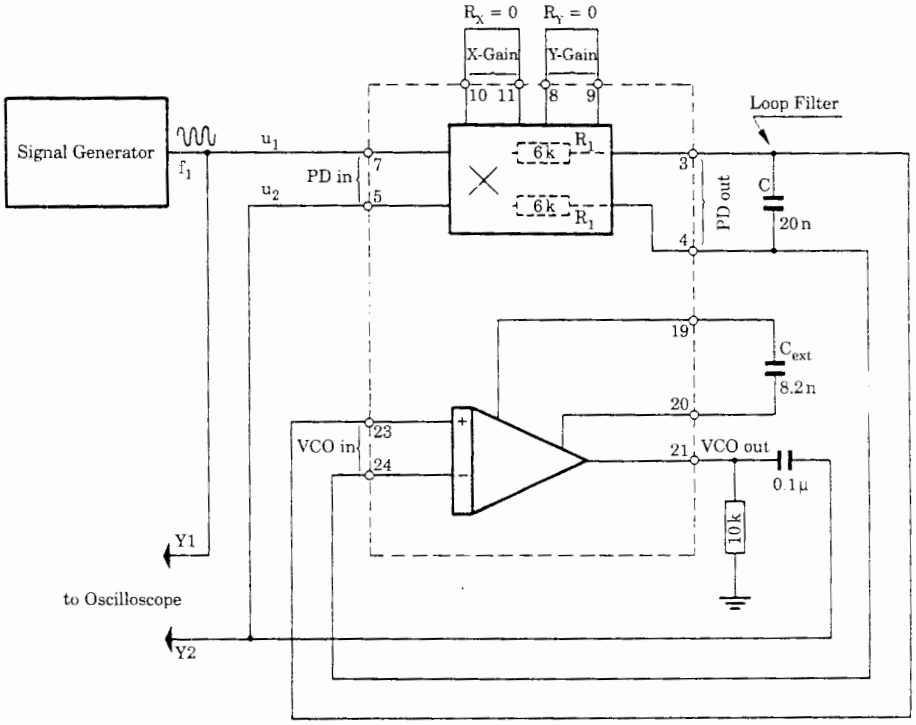


Figure 11.7 Test circuit for the measurement of hold range $\Delta\omega_H$ and pull-in range $\Delta\omega_P$.

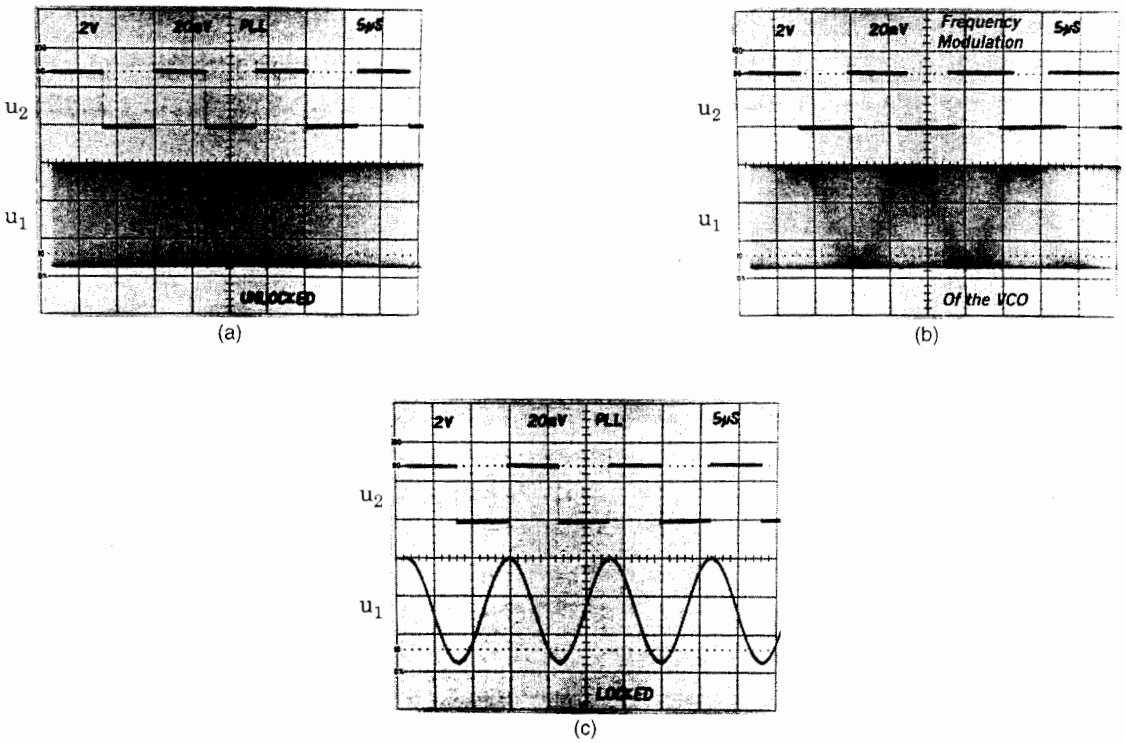


Figure 11.8 Waveforms of signals u_1 and u_2 in the test circuit of Fig. 11.7. (*a*) PLL unlocked. (*b*) PLL near locking. (*c*) PLL locked.

Determination of the hold range Δf_H and the pull-in range Δf_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 Δf_P is determined in a similar way. To get Δf_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, 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 Δf_P.

11.5 Measurement of Natural Frequency ω_n, Damping Factor ζ, and Lock Range $\Delta\omega_L$

The PLL circuit of Fig. 11.7 is used for the following measurements. To measure the natural frequency ω_n and the damping factor ζ of a PLL, we apply a disturbance to the PLL that 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. 11.9. The PLL under test 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 much smaller than the center frequency, for example, 1 kHz.

If the frequency of the signal generator is abruptly changed, a phase error θ_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 analyzed by recording on an oscilloscope. This is shown by the waveforms in Fig. 11.10*a*, 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.

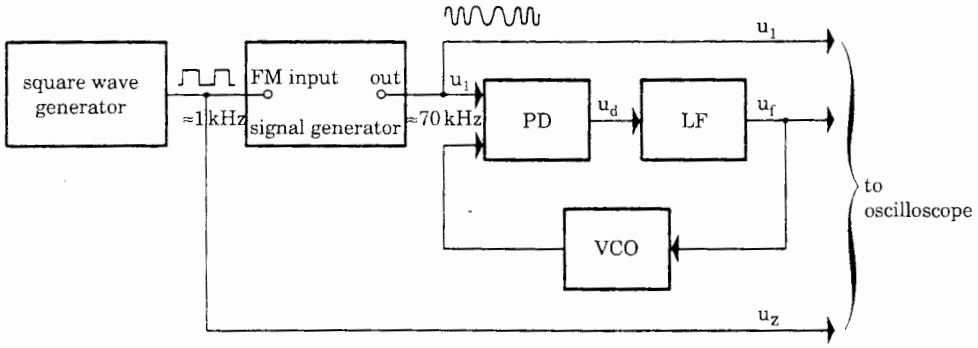


Figure 11.9 Test circuit for the measurement of natural frequency ω_n and damping factor ζ.

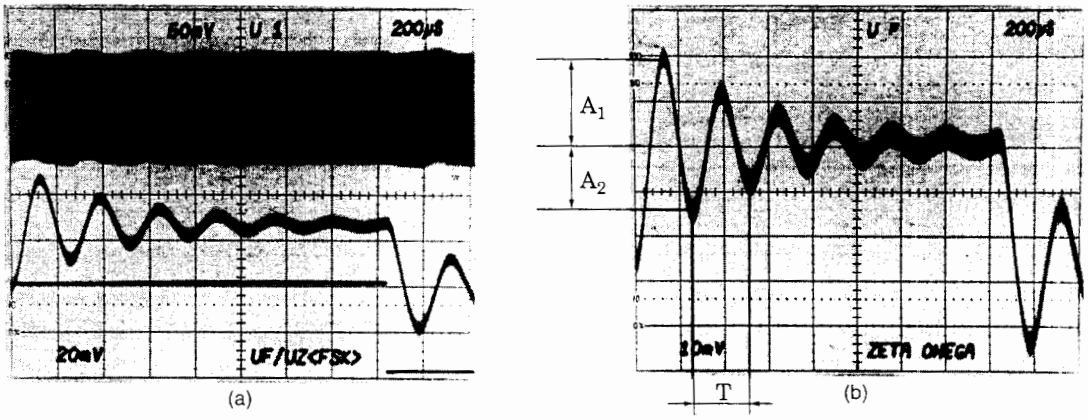


Figure 11.10 Waveforms of the test circuit shown in Fig. 11.9. (*a*) Top trace—input signal u_1; center trace—output signal of the loop filter; bottom trace—modulating signal, 1-kHz square wave. (*b*) Enlarged view of the u_f waveform shown in (*a*), center trace.

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 ω_n and ζ can be calculated from the waveform of u_f. Figure 11.10*b* is an enlarged view of this signal; ζ 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

$$\zeta = \frac{\ln(A_1/A_2)}{\left[\pi^2 + \{\ln(A_1/A_2)\}^2\right]^{1/2}}$$

The natural frequency ω_n is calculated from the period T of one oscillation in Fig. 11.10*b* according to

$$\omega_n = \frac{2\pi}{T\sqrt{1-\zeta^2}}$$

Let us now evaluate numerically the waveform of Fig. 11.10*b*. For A_1 and A_2 we read approximately 1.9 and 1.5 divisions, respectively. Consequently we obtain

$$\zeta \approx 0.08$$

For T we find $T \approx 240\,\mu s$. Hence ω_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 us calculate the values of ω_n and ζ using the theory of the linear PLL [Eq. (2.41)] and check whether our measurements agree with the predicted results. Using Eq. (11.1), 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. 11.10*a* 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 $K_d \approx 3.7$ V/rad from Fig. 11.6. The two time constants τ_1 and τ_2 can be derived from the test circuit in Fig. 11.7:

$$\tau_1 = 12 k\Omega \cdot 20 \text{ nF} = 240 \mu\text{s}$$
$$\tau_2 = 0$$

Using Eq. (2.41), we finally get

$$\omega_n = \left(\frac{K_0 K_d}{N(\tau_1 + \tau_2)} \right)^{1/2} = \left(\frac{51.1 \cdot 10^3 \cdot 3.7}{240 \cdot 10^{-6}} \right)^{1/2} = 28 \cdot 10^3 \text{ rad s}^{-1}$$
$$\zeta = \frac{\omega_n}{2} \left(\tau_2 + \frac{N}{K_0 K_d} \right) = \frac{28 \cdot 10^3}{2 \cdot 51.1 \cdot 10^3 \cdot 3.7} = 0.07$$

(*Note*: $N = 1$.) This agrees well with the experimental measurements.

Note that this experimental method of measuring ω_n and ζ is applicable only for $\zeta < 1$. This requirement is met, however, in most cases. If ζ were greater than 1, the transient response would become aperiodic, and it would become impossible to define the values A_1 and A_2 in Fig. 11.10*a*. In the example of Fig. 11.10*a* the damping factor ζ has purposely been chosen 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 ζ, τ_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. 11.9, 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 ω_0. The higher of the two frequencies (ω_{high}) generated by the signal generator must now be chosen higher than the pull-out frequency, i.e., $\omega_{high} > \omega_0 + \Delta\omega_P$. The lower frequency ($\omega_{low}$) has to be chosen initially to be as high as the higher ($\omega_{low} = \omega_{high}$). In this situation the amplitude of the square wave generator will be zero. The test circuit must now be tuned such that ω_{high} stays always the same, but that ω_{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. 11.10*a*. During the interval where the frequency of the signal generator is high, u_f will be a "high-frequency" signal, which is visible only as a smeared trace. When

the lower frequency reaches a value around $\omega_{low} \approx \omega_0 + \Delta\omega_L$, the PLL will lock quickly; i.e., 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 $\omega_{low} - \omega_0$. For larger values of ω_{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.

11.6 Measurement of the Phase-Transfer Function $H(\omega)$ and the 3-dB Bandwidth ω_{3dB}

There are different methods of measuring $H(\omega)$; some of these use PM techniques, others use frequency modulation.[35,49] The PM technique has the advantage that the maximum phase error θ_e can be limited to small values (e.g., $\theta_e < \pi/4$) so the PD never operates in its nonlinear region. Unfortunately, most signal generators used in the lab can be frequency-modulated but not phase-modulated.

To enable the measurement of $H(\omega)$, we use FM techniques, therefore. The test setup is shown in Fig. 11.11. The frequency of the signal generator is tuned to the center frequency of the PLL. A sine wave is applied now to the FM input. Consequently, the output signal of the generator is given by

$$\omega_1 = \omega_0 + \Delta\omega \sin \omega_m t \tag{11.2}$$

where ω_m is the modulating frequency and $\Delta\omega$ is the peak frequency deviation. The size of $\Delta\omega$ will be determined later. Because the phase $\theta_1(t)$ of the reference signal is equal to the integral of the frequency deviation $\Delta\omega \sin(\omega_m t)$ over time, we have

$$\theta_1(t) = -\frac{\Delta\omega}{\omega_m} \cos \omega_m t \tag{11.3}$$

The amplitude of the reference phase signal $\theta_1(t)$ is therefore

$$|\Theta_1(\omega_m)| = \frac{\Delta\omega}{\omega_m} \tag{11.4}$$

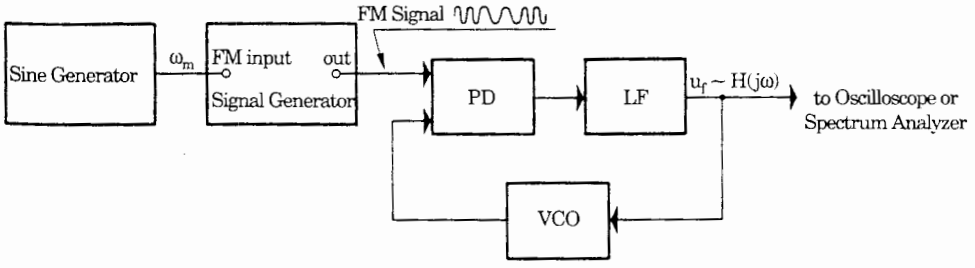


Figure 11.11 Test circuit for the measurement of $H(\omega)$.

Appendix

A

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, i.e., for a LPLL that contains a passive loop filter.[1,2,4] 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 a DPLL using a PFD. As pointed out in Sec. 2.6.2, under "Pull-in Range $\Delta\omega_P$ and Pull-in Time T_P,"the model of Fig. 2.39 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.

A.1 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 ω_1 of the reference signal is furthermore assumed to be so much higher than the (scaled-down) center frequency ω_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 ω_0. As we know from Sec. 2.6.2, 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 ω_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

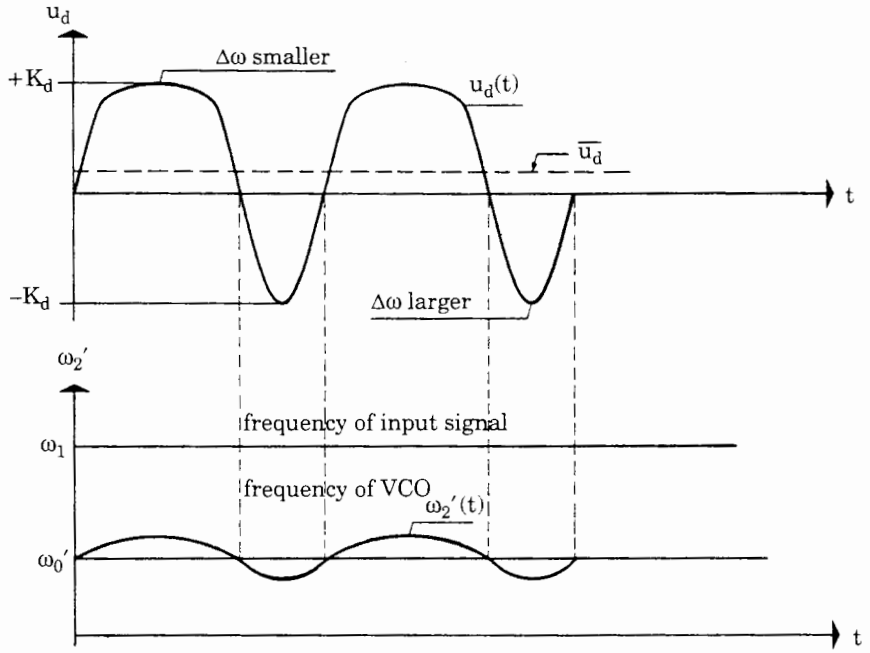


Figure A.1 Plot of $u_d(t)$ and $\omega_2'(t)$ against time for the unlocked PLL. It is assumed that the power supply of the PLL has been turned on at $t = 0$.

output signal is *not zero*, but slightly *positive*. This slowly pulls the frequency ω_2 of the VCO in the positive direction.

Under certain conditions to be discussed in the following, this process is *regenerative*; i.e., the down-scaled VCO output frequency ω_2' is pulled to a frequency close enough to ω_1 that the PLL finally locks. For the calculation of the pull-in process, we assume that the down-scaled frequency ω_2' of the VCO has already been pulled somewhat in the direction of ω_1 (Fig. A.2). The average down-scaled angular frequency of the VCO is denoted by ω_{20}', and the average frequency offset is $\Delta\omega = \omega_1 - \omega_{20}'$. Next the average value $\overline{u_d}$ is calculated as a function of the frequency offset $\Delta\omega$.

As will be demonstrated below, $\overline{u_d}$ 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 $\omega_2'(t)$ (see the lower curve in Fig. A.2) are known exactly. At points A and C the instantaneous frequency $\omega_2'(t)$ is exactly ω_{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

$$\omega_2'(t) = \omega_{20}' - F_H K_0 K_d / N$$

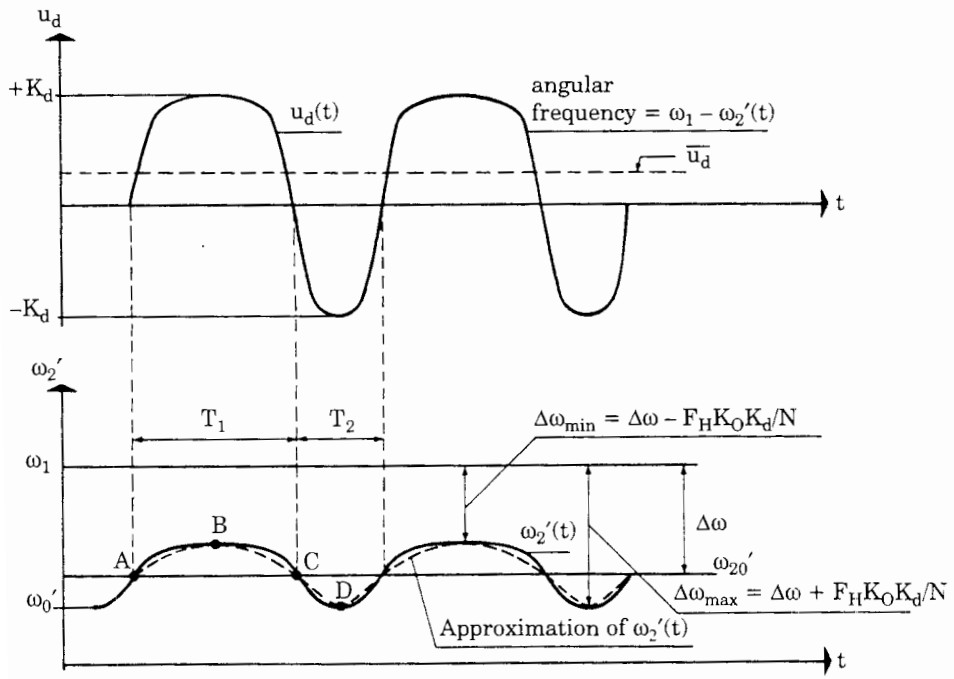


Figure A.2 Plot of $u_d(t)$ and $\omega_2'(t)$ for the unlocked PLL during the pull-in process. It is assumed that the frequency of the VCO has already been pulled somewhat toward the reference frequency ω_1.

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 now to 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 $2/\pi$.

For $\overline{\omega_{2+}'}$ and $\overline{\omega_{2-}'}$ we get

$$\overline{\omega_{2+}'} = \omega_{20}' + \frac{2}{\pi} F_H K_0 K_d / N \tag{A.5a}$$

$$\overline{\omega_{2-}'} = \omega_{20}' - \frac{2}{\pi} F_H K_0 K_d / N \tag{A.5b}$$

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:

$$\overline{\Delta\omega_+} = \Delta\omega - \frac{2}{\pi} F_H K_0 K_d / N \tag{A.6a}$$

$$\overline{\Delta\omega_-} = \Delta\omega + \frac{2}{\pi} F_H K_0 K_d / N \tag{A.6b}$$

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 $\overline{\Delta\omega_-}$:

$$T_1 = \frac{1}{2}\frac{2\pi}{\overline{\Delta\omega_+}} = \frac{\pi}{\Delta\omega - (2/\pi)F_H K_d K_0/N} \tag{A.7a}$$

$$T_2 = \frac{1}{2}\frac{2\pi}{\overline{\Delta\omega_-}} = \frac{\pi}{\Delta\omega + (2/\pi)F_H K_d K_0/N} \tag{A.7b}$$

Knowing T_1 and T_2, we can now calculate $\overline{u_d}$. As shown in the upper curve 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 ω_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 $\overline{u_d}$; 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

$$\overline{u_d} = \frac{2}{\pi}K_d\frac{T_1-T_2}{T_1+T_2} \tag{A.8}$$

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:

$$T_1 = \frac{\pi}{\Delta\omega\left[1-\frac{2}{\pi}\frac{F_H K_d K_0}{\Delta\omega N}\right]} \approx \frac{\pi}{\Delta\omega}\left(1+\frac{2}{\pi}\frac{F_H K_d K_0}{\Delta\omega N}\right) \tag{A.9a}$$

$$T_2 = \frac{\pi}{\Delta\omega\left[1+\frac{2}{\pi}\frac{F_H K_d K_0}{\Delta\omega N}\right]} \approx \frac{\pi}{\Delta\omega}\left(1-\frac{2}{\pi}\frac{F_H K_d K_0}{\Delta\omega N}\right) \tag{A.9b}$$

With these approximations $\overline{u_d}$ becomes

$$\overline{u_d} = \frac{4F_H K_d^2 K_0}{\pi^2 \Delta\omega N} \tag{A.10}$$

This can be written in simplified form as

$$\overline{u_d} = \frac{c_d}{\Delta\omega} \tag{A.10a}$$

with

$$c_d = \frac{4F_H K_d^2 K_0}{\pi^2 N}$$

Thus the average phase detector output signal $\overline{u_d}$ is inversely proportional to $\Delta\omega$. This is valid for large values of $\Delta\omega$ only, because $\overline{u_d}$ can never become larger than K_d, as is easily seen from Fig. A.2. If the average u_d signal is plotted against $\Delta\omega$ (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 $\overline{u_d}$ as a function of the frequency offset $\Delta\omega$. As defined by Eq. (1.1), the (down-scaled) average frequency of $\overline{\omega_{20}'}$ of the VCO is given by

$$\overline{\omega_{20}'} = \omega_0 + \frac{K_0}{N}\overline{u_f} \tag{A.11}$$

where $\overline{u_f}$ 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

$$\Delta\omega = \Delta\omega_0 - \frac{K_0}{N}\overline{u_f}$$

or

$$\overline{u_f} = \frac{\Delta\omega_0 - \Delta\omega}{K_0/N} \tag{A.12}$$

Plotting $\overline{u_f}$ 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.

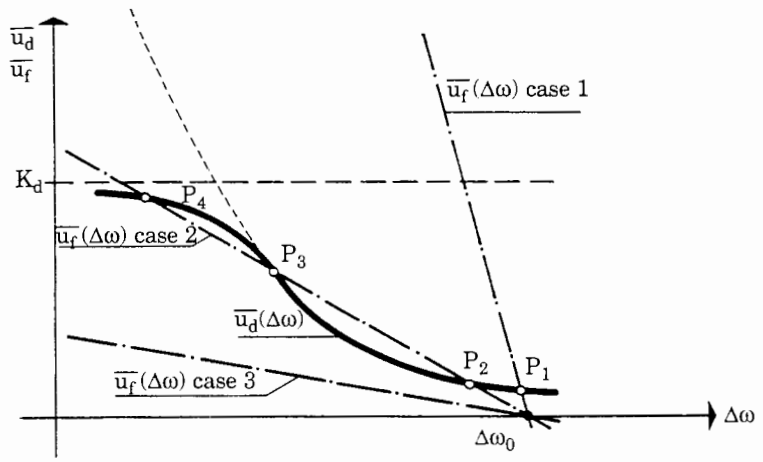


Figure A.3 Plot of the average signals $\overline{u_d}(t)$ and $\overline{u_f}(t)$ against frequency offset $\Delta\omega$ for the unlocked PLL.

In case 2 the curves $\overline{u_d}\ (\Delta\omega)$ and $\overline{u_f}\ (\Delta\omega)$ 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

$$\overline{u_f}(\Delta\omega) = \overline{u_d}(\Delta\omega) \tag{A.13}$$

does not have a real solution.

Inserting Eqs. (A.10a) and (A.12) into (A.13) yields the quadratic equation

$$\Delta\omega^2 - \Delta\omega_0\Delta\omega + \frac{K_0}{N}c_d = 0 \tag{A.14}$$

Its solutions are

$$\Delta\omega_{1,2} = \frac{\Delta\omega_0 \pm \sqrt{\Delta\omega_0^2 - 4\dfrac{K_0}{N}c_d}}{2\dfrac{K_0}{N}c_d} \tag{A.15}$$

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

$$\Delta\omega_P = 2\sqrt{\frac{K_0}{N}c_d} \tag{A.16}$$

If we insert c_d from Eq. (A.10a) and make use of the substitutions defined in Eqs. (2.41) and (2.42), we finally get the expressions

$$\Delta\omega_P = \frac{4}{\pi}\sqrt{2\zeta\omega_n \frac{K_0}{N} K_d - \omega_n^2} \tag{A.17a}$$

for a passive lead-lag filter and a low-gain loop,

$$\Delta\omega_P = \frac{4\sqrt{2}}{\pi}\sqrt{\zeta\omega_n \frac{K_0}{N} K_d} \tag{A.17b}$$

for a passive lead-lag filter and a high-gain loop,

$$\Delta\omega_P = \frac{4}{\pi}\sqrt{2\zeta\omega_n \frac{K_0}{N} K_d - \frac{\omega_n^2}{K_a}} \tag{A.18a}$$

for an active lead-lag filter and a low-gain loop, and

$$\Delta\omega_P = \frac{4\sqrt{2}}{\pi}\sqrt{\zeta\omega_n \frac{K_0}{N} K_d} \tag{A.18b}$$

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. 2.3.6.

We conclude that the model shown in Fig. A.3 enables us to calculate the pull-in range $\Delta\omega_P$. To get that result, we simply postulated that the two curves for $\overline{u_d}(\Delta\omega)$ and $\overline{u_f}(\Delta\omega)$ 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.2 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 which 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 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 $\overline{u_d}$ varies in inverse proportion to the frequency offset $\Delta\omega = \omega_1 - \omega_{20}'$, where ω_{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

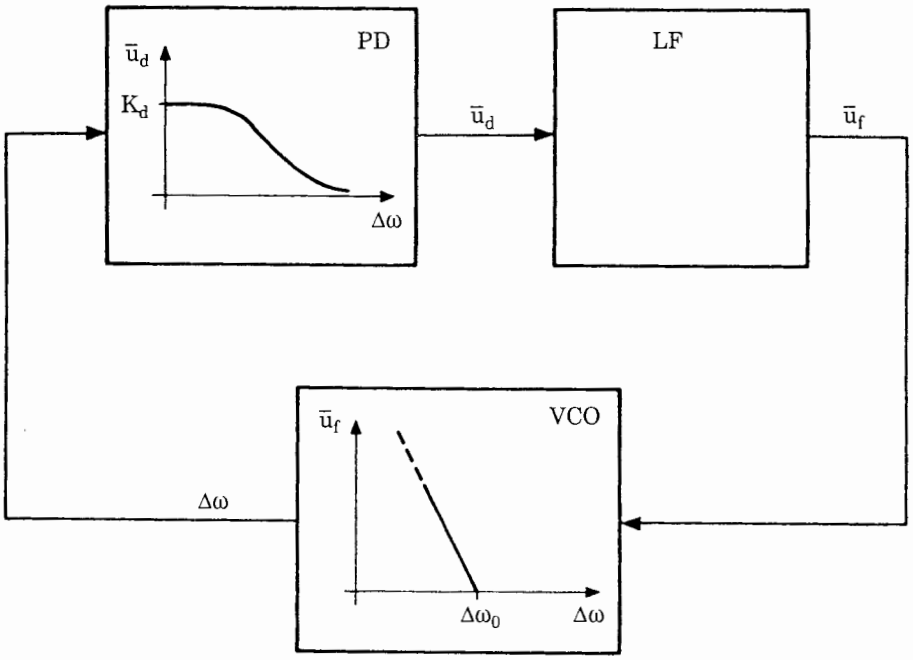


Figure A.4 Mathematical model for the pull-in process of the PLL.

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

$$F(s) = \frac{1 + s\tau_2}{1 + s(\tau_1 + \tau_2)} \tag{A.19a}$$

When transformed back into time domain, we have

$$\overline{u_f}(t) + (\tau_1 + \tau_2)\frac{d}{dt}\overline{u_f}(t) = \overline{u_d}(t) + \tau_2 \frac{d}{dt}\overline{u_d}(t) \tag{A.19b}$$

The three equations (A.10*a*), (A.12), and (A.19*b*) fully describe the pull-in process. When $\overline{u_d}$ and $\overline{u_f}$ are eliminated, we get the nonlinear differential equation

$$\frac{d}{dt}\Delta\omega\left(\frac{\tau_1 + \tau_2}{K_0/N} - \frac{\tau_2 c_d}{\Delta\omega^2}\right) + \frac{c_d}{\Delta\omega} + \frac{\Delta\omega}{K_0/N} = \frac{\Delta\omega_0}{K_c/N} \tag{A.20}$$

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

$$F(s) = \frac{1}{s\tau_1} \tag{A.21a}$$

which is the transfer function of an ideal integrator. The following discussion shows under which circumstances this approximation is acceptable.

Figure A.5*a* shows the step response $g(t)$ of a passive lag filter having the transfer function given in Eq. (A.19*a*) for the case $\tau_1 >> \tau_2$. In Fig. A.5*b*, the step response $g(t)$ of the ideal integrator [Eq. (A.21*a*)] 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 τ_1. Experience shows that this is not necessarily fulfilled. In cases where T_P turns out to be less than τ_1, the approximation is quite good; i.e., the errors are normally below

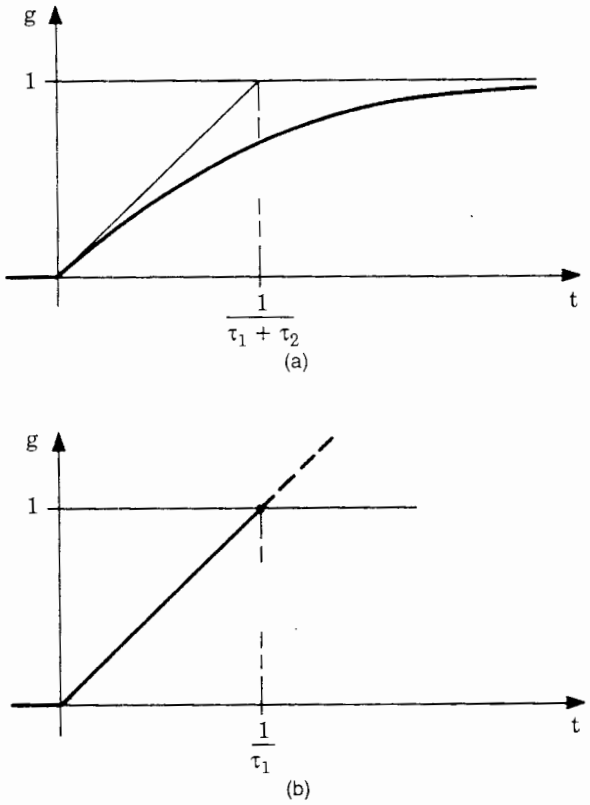


Figure A.5 Step response of first-order low-pass filters. (*a*) Step response of a passive lag filter. (*b*) Step response of the ideal integrator.

10 percent of the predicted value. When T_P turns out to be larger than τ_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 below, delivers reasonable estimates (errors less than 10 percent) when $\Delta\omega_0$ is less than about 0.8 $\Delta\omega_P$.

Transforming Eq. (A.21*a*) back into time domain yields

$$\overline{u_d}(t) = \tau_1 \frac{d}{dt} \overline{u_f}(t) \tag{A.21b}$$

Inserting Eqs. (A.10*a*) and (A.21*b*) into Eq. (A.12) yields the simplified differential equation

$$\frac{d}{dt}\Delta\omega \frac{\tau_1}{K_0/N} + \frac{c_d}{\Delta\omega} = 0 \tag{A.22}$$

Although this equation is still nonlinear, the variables $\Delta\omega$ and t can be separated, so we have

$$\frac{\tau_1}{c_d K_0/N} \int_{\Delta\omega_0}^{\Delta\omega_L} \Delta\omega \, d\,\Delta\omega = -\int_0^{T_P} dt \tag{A.23}$$

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, as has been shown by the simulations in Sec. 2.6.2. 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

$$T_P \approx \frac{\tau_1 \Delta\omega_0^2}{2c_d K_0/N} \tag{A.24}$$

Inserting c_d from Eq. (A.10*a*) and using the substitutions of Eq. (2.15*a*), we get for the pull-in time

$$T_P \approx \frac{\pi^2}{16} \frac{\Delta\omega_0^2}{\zeta \omega_n^3} \tag{A.25a}$$

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 becomes

$$T_P \approx \frac{\pi^2}{16} \frac{\Delta\omega_0^2 K_a}{\zeta \omega_n^3} \tag{A.25b}$$

Finally, when the loop filter is an active PI filter, the pull-in time is given by

$$T_P \approx \frac{\pi^2}{16\,\zeta}\frac{\Delta\omega_0^2}{\omega_n^3} \tag{A.25c}$$

A.3 The Pull-in Range $\Delta\omega_P$ of the DPLL

The approach described in Sec. A.1 can be adopted to calculate also the pull-in range of the digital PLL. We assume first that the EXOR gate is used as phase detector. As explained in Sec. 2.6.2. and Fig. 2.36, 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 $\overline{u_d}$. The result is

$$\overline{u_d} = \frac{\pi^2 K_d^2 K_0 F_H}{16\,\Delta\omega\, N} \tag{A.26a}$$

which can be rewritten as

$$\overline{u_d} = \frac{c_d}{\Delta\omega} \tag{A.26b}$$

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 LPLL, so the pull-in range is obtained simply by inserting c_d into Eq. (A.16). The result becomes

$$\Delta\omega_P = \frac{\pi}{2}\sqrt{2\zeta\omega_n K_0 K_d/N - \omega_n^2} \tag{A.27a}$$

(PD = EXOR, passive lead-lag filter, low-gain loop),

$$\Delta\omega_P = \frac{\pi}{\sqrt{2}}\sqrt{\zeta\omega_n K_0 K_d/N} \tag{A.27b}$$

(PD = EXOR, passive lead-lag filter, high-gain loop),

$$\Delta\omega_P = \frac{\pi}{2}\sqrt{2\zeta\omega_n K_0 K_d/N - \frac{\omega_n^2}{K_a}} \tag{A.28a}$$

(PD = EXOR, active lead-lag filter, low-gain loop), and

$$\Delta\omega_P = \frac{\pi}{\sqrt{2}}\sqrt{\zeta\omega_n K_0 K_d / N} \tag{A.28b}$$

(PD = EXOR, active lead-lag filter, high-gain loop).

The pull-in range for a DPLL using a JK-flipflop phase detector remains to be calculated. As shown in Sec. 2.6.2. and Fig. 2.37, the phase detector output signal is an unsymmetrical sawtooth in the unlocked state. Again using the same argument as in Sec. A.1, the coefficient c_d can be shown to be

$$c_d = \frac{\pi^2 K_d^2 K_0 F_H}{4N}$$

in this case. If we insert c_d into Eq. (A.16) and make use of the substitutions in Eq. (2.42), we get

$$\Delta\omega_P = \pi\sqrt{2\zeta\omega_n K_0 K_d / N - \omega_n^2} \tag{A.29a}$$

(PD = *JK*-flipflop, passive lag filter, low-gain loop)

$$\Delta\omega_P = \pi\sqrt{2\zeta\omega_n K_0 K_d / N} \tag{A.29b}$$

(PD = *JK*-flipflop, passive lag filter, high-gain loop)

$$\Delta\omega_P = \pi\sqrt{2\zeta\omega_n K_0 K_d / N - \frac{\omega_n^2}{K_a}} \tag{A.30a}$$

(PD = *JK*-flipflop, active lag filter, low-gain loop)

$$\Delta\omega_P = \pi\sqrt{2\zeta\omega_n K_0 K_d / N} \tag{A.30b}$$

(PD = *JK*-flipflop, active lag filter, high-gain loop).

A.4 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:

$$\overline{u_d} = \frac{c_d}{\Delta\omega}$$

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 [for the substitutions of Eqs. (2.41) to (2.43)]

$$T_P \approx \frac{4}{\pi^2} \frac{\Delta\omega_0^2}{\zeta\, \omega_n^3} \tag{A.31a}$$

for the passive lead-lag filter,

$$T_P \approx \frac{4}{\pi^2} \frac{\Delta\omega_0^2 K_a}{\zeta\, \omega_n^3} \tag{A.31b}$$

for the active lead-lag filter, and

$$T_P \approx \frac{4}{\pi^2} \frac{\Delta\omega_0^2}{\zeta\, \omega_n^3} \tag{A.31c}$$

for the active PI filter.

When the DPLL uses the JK-flipflop phase detector, the pull-in time T_P becomes [for the substitutions in Eqs. (2.41) to (2.43)]

$$T_P \approx \frac{1}{\pi^2} \frac{\Delta\omega_0^2}{\zeta\, \omega_n^3} \tag{A.32a}$$

for the passive lead-lag filter,

$$T_P \approx \frac{1}{\pi^2} \frac{\Delta\omega_0^2 K_a}{\zeta\, \omega_n^3} \tag{A.32b}$$

for the active lead-lag filter, and

$$T_P \approx \frac{1}{\pi^2} \frac{\Delta\omega_0^2}{\zeta\, \omega_n^3} \tag{A.32c}$$

for the active PI filter.

As we stated in Sec. 2.6.2, the approximations for the pull-in time are valid if the initial frequency offset $\Delta\omega_0$ is markedly below the pull-in range $\Delta\omega_P$, i.e., below about 0.8 $\Delta\omega_P$. If the initial frequency offset comes close to $\Delta\omega_P$, the pull-in time approaches infinity.

Appendix

B

The Laplace Transform

B.1 Transforms Are the Engineer's Tools

Trying to solve electronic problems without using the Laplace transform is like 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). 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 ω_0. This square wave can now be thought to be composed of a(n) (infinite) number of sine wave signals having frequencies ω_0 (fundamental frequency), $2\omega_0$, $3\omega_0, \ldots$ (harmonics). Mathematically, any periodic signal $f(t)$ having a repetition frequency of ω_0 can be written as a sum of its harmonics,

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.

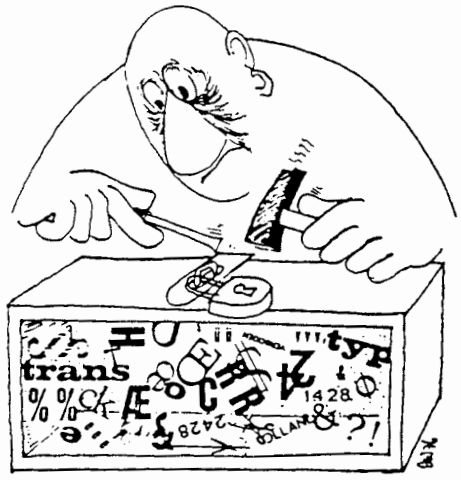


Figure B.2 Looking at the transient response of electric networks without using the Laplace transform can be tricky . . .

$$f(t) = \sum_{n=-\infty}^{+\infty} F(n\omega_0) \exp(jn\omega_0 t) \tag{B.1}$$

The so-called Fourier coefficients $F(n\omega_0)$ are calculated from

$$F(n\omega_0) = \frac{1}{T} \int_{-T/2}^{T/2} f(t) \exp(-jn\omega_0 t)\, dt \tag{B.2}$$

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

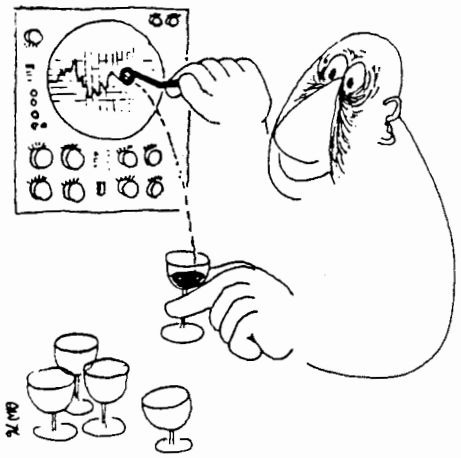


Figure B.3 ... but the engineer familiar with Laplace techniques may find the solution very quickly.

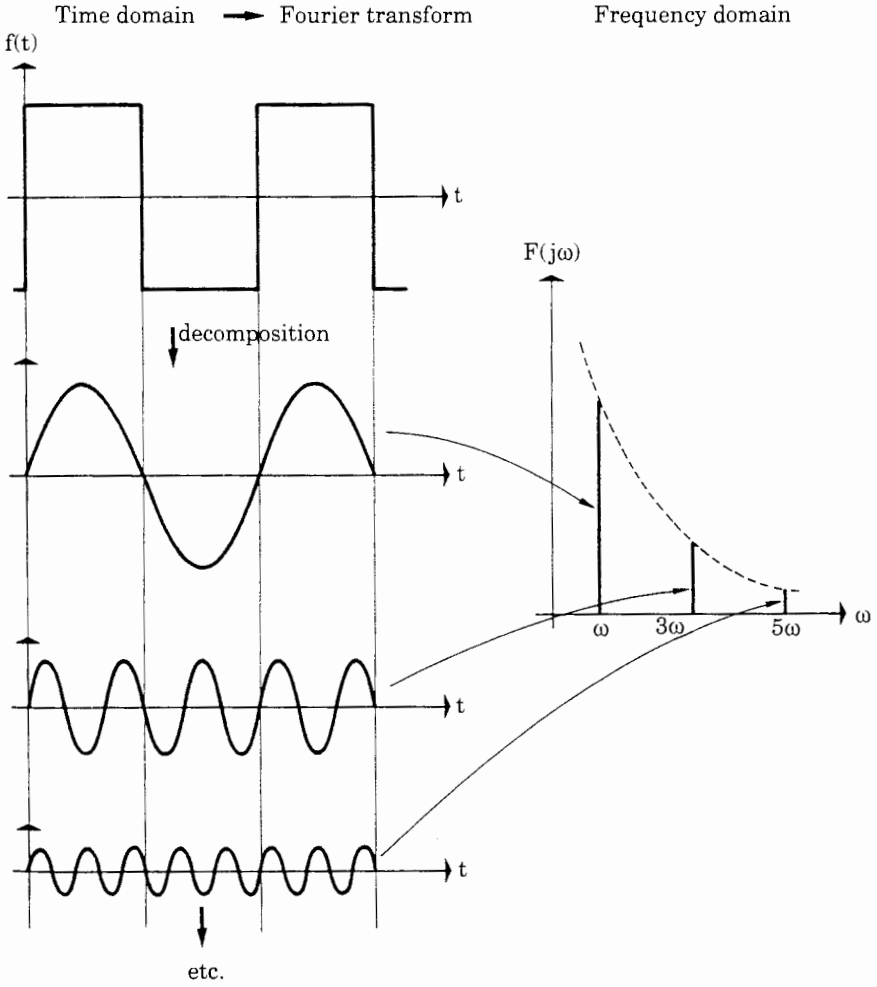


Figure B.4 The principle of the Fourier transform.

Fourier coefficients $F(n\omega_0)$ are generally complex numbers; we can therefore write

$$F(n\omega_0) = |F(n\omega_0)| \exp(j\Phi_n) \tag{B.3}$$

where $|F(n\omega_0)|$ is the amplitude and Φ_n the phase of $F(n\omega_0)$.

In plotting the Fourier transform of a signal $f(t)$, the amplitude $|F(n\omega_0)|$ and the phase Φ_n are normally plotted against ω; 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 ω_0, $2\omega_0$, $3\omega_0$, ...

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 shows only lines at ω_0, $3\omega_0$, $5\omega_0$, ...

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,

$$f(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega) e^{j\omega t} dt \tag{B.4}$$

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 ∞ in this equation, we get

$$F(\omega) = \int_{-\infty}^{\infty} f(t) e^{-j\omega t} dt \tag{B.5}$$

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 are tabulated in many reference books.[36]

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 $f(t) = \sin \omega t$. The Laplace transform offers a way to circumvent this problem.

B.2 A 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

$$F(\omega) = \int_0^\infty f(t)e^{-j\omega t}\,dt = \int_0^\infty \sin\omega_0 t\; e^{-j\omega t}\,dt$$

Using an integral table, we find for $F(\omega)$

$$F(\omega) = \frac{-j\omega e^{-j\omega t}\sin\omega_0 t - \omega_0 e^{-j\omega t}\cos\omega_0 t}{\omega_0^2 - \omega^2}\Bigg|_0^\infty$$

Introducing the limits of integration (here 0 and ∞) 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 σ is a positive real number:

$$F'(\omega,\sigma) = \int_0^\infty [f(t)e^{-\sigma t}]e^{-j\omega t}\,dt \tag{B.6}$$

The notation $F'(\omega,\sigma)$ is chosen to emphasize that now F' is also dependent on σ. 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 σ is chosen larger than a, the product approaches zero for $t \to \infty$. 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

$$F'(\omega,\sigma) = \frac{-(\sigma + j\omega)e^{-(\sigma+j\omega)t}\sin\omega_0 t - \omega_0 e^{-(\sigma+j\omega)t}\cos\omega_0 t}{\omega_0^2 + (\sigma + j\omega)^2}$$

If the limits of integration are now inserted, the term $e^{-(\sigma+j\omega)\infty}$ becomes zero for positive σ. Then $F'(\omega,\sigma)$ becomes

$$F'(\omega,\sigma) = \frac{\omega_0}{\omega_0^2 + (\sigma + j\omega)^2}$$

Of course, $F'(\omega,\sigma)$ contains the damping factor σ. The Fourier integral $F(\omega)$ is now obtained simply by letting $\sigma \to 0$:

$$F(\omega) = \lim_{\sigma \to 0} F'(\omega,\sigma) = \frac{\omega_0}{\omega_0^2 - \omega^2}$$

If σ 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 $\sigma \to 0$.*

Let us 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 σ and ω 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, $F(\omega)$ is a *complex* function of the *real* variable ω. To plot $F(\omega)$ against ω, we have to plot amplitude $|F(\omega)|$ and phase $\Phi(\omega)$ as a function of ω. (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 *zeros* 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 (●) or zeros (○). [Note that not every function $F(s)$ necessarily has poles or zeros.] 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 $\sigma = 0$:

$$F(\omega) = \lim_{\sigma \to 0} F(s), \qquad s = \sigma + j\omega \tag{B.8}$$

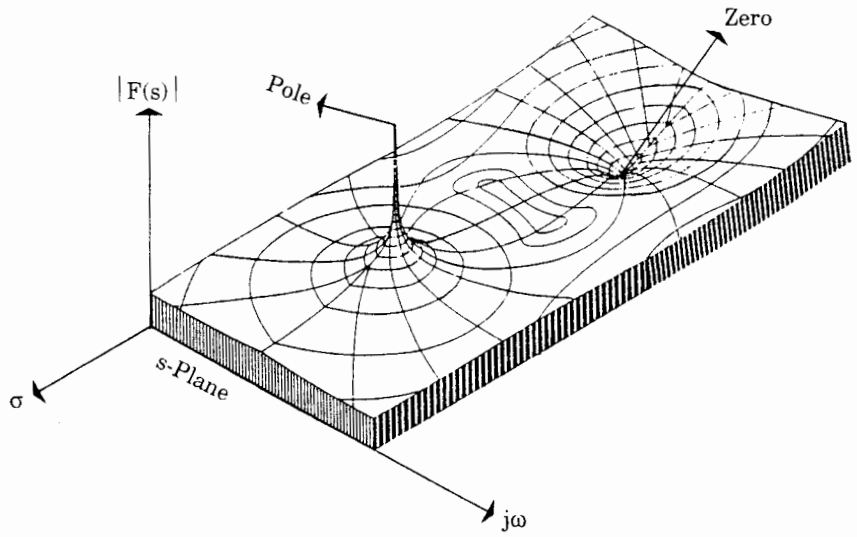


Figure B.5 The magnitude of the Laplace transform $F(s)$ plotted as a function of complex frequency s yields a relief. This illustration shows a pole and a zero of the transfer function $F(s)$.

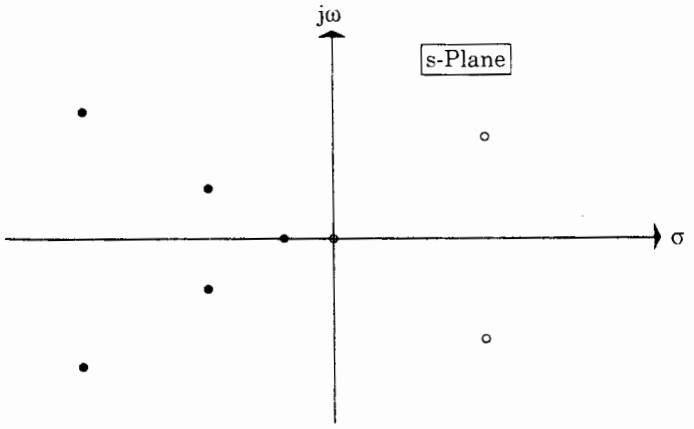


Figure B.6 Plot of the poles and zeros of a given transfer function $F(s)$.

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 is also possible to calculate $f(t)$ if $F(s)$ is given first. This transformation is called *inverse* Laplace transform and is defined by

$$f(t) = \frac{1}{2\pi j} \int_{c-j\infty}^{c+j\infty} F(s)\, e^{-st}\, ds \tag{B.9}$$

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, let us introduce some convenient abbreviations:

$$F(s) = L\{f(t)\} \tag{B.10a}$$

$$f(t) = L^{-1}\{F(s)\} \tag{B.10b}$$

Equation (B.10a) says that $F(s)$ is the Laplace transform of the signal $f(t)$; Eq. (B.10b) states that $f(t)$ is the inverse Laplace transform of $F(s)$.

B.3 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 us 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),

$$F(s) = \int_0^\infty u(t) e^{-st} dt = \int_0^\infty e^{-st} dt = \left. \frac{e^{-st}}{-s} \right|_0^\infty$$

$$= -\frac{1}{s}(e^{-\infty} - e^0) = -\frac{1}{s}(0 - 1) = \frac{1}{s}$$

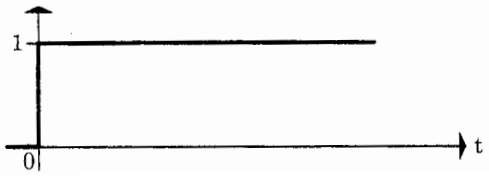


Figure B.7 Plot of the unit step function against time.

that is,

$$F(s) = \frac{1}{s}$$

$F(s)$ has one single pole at $s = 0$, that is, 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.

B.4 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 below.

B.4.1 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 $F_2(s)$:

$$L\{f_1(t) + f_2(t)\} = L\{f_1(t)\} + L\{f_2(t)\} = F_1(s) + F_2(s) \tag{B.11}$$

B.4.2 Multiplication by a constant factor *k*

For the same reason we have

$$L\{kf(t)\} = kL\{f(t)\} = kF(s) \tag{B.12}$$

If the signal $f(t)$ is multiplied by a constant factor k, the Laplace transform $F(s)$ is simply multiplied by the same factor.

B.4.3 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 transforms $F_1(s) = \mathrm{L}\{f_1(t)\}$ and $F_2(s) = \mathrm{L}\{f_2(t)\}$. 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*.[3,13,14,37] We will not consider this operation in detail here, but will define it for completeness:

$$L\{f_1(t) \cdot f_1(t)\} = \frac{1}{2\pi j} \oint F_1(\chi) \cdot F_2(s - \chi)\, d\chi \tag{B.13}$$

TABLE B.1 Laplace Transform, $f(t) = 0$ for $t < 0$

	Signal $f(t)$	Laplace transform $F(s)$
General rules		
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	$f_1(t) \cdot f_2(t)$	$F_1(s) * F_2(s)$
	$f_1(t) * f_2(t)$	$F_1(s) \cdot F_2(s)$
Functions		
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 fractional)
Exponential functions		
	$e^{\alpha t}$	$\frac{1}{s-\alpha}$
	$\frac{1}{\alpha}(e^{\alpha t} - 1)$	$\frac{1}{s(s-\alpha)}$
	$\frac{1}{\alpha}(1 - e^{-\alpha t})$	$\frac{1}{s(s+\alpha)}$
	$\frac{t^n - 1}{(n-1)!e^{\alpha t}}$	$\frac{1}{(s-\alpha)^n}, n > 0$ (can be fractional)
Trigonometric functions		
	$\frac{1}{\alpha}\sin\alpha t$	$\frac{1}{s^2+\alpha^2}$
	$\cos\alpha t$	$\frac{s}{s^2+\alpha^2}$
	$\frac{1}{\alpha^2}(1 - \cos\alpha t)$	$\frac{1}{s(s^2+\alpha^2)}$

TABLE B.1 *Continued*

	Signal $f(t)$	Laplace transform $F(s)$
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)		
	$\frac{e^{-\zeta\omega_n t}\sin\sqrt{1-\zeta^2}\omega_n t}{\sqrt{1-\zeta^2}\omega_n}$	$\frac{1}{s^2+2s\zeta\omega_n+\omega_n^2}$
	$\left[\cos\sqrt{1-\zeta^2}\omega_n t-\frac{\zeta}{\sqrt{1-\zeta^2}}\sin\sqrt{1-\zeta^2}\omega_n t\right]e^{-\zeta\omega_n t}$	$\frac{s}{s^2+2s\zeta\omega_n+\omega_n^2}$
	$\left[\frac{2\zeta^2-1}{\sqrt{1-\zeta^2}}\sin\sqrt{1-\zeta^2}\omega_n t-2\zeta\cos\sqrt{1-\zeta^2}\omega_n t\right]\omega_n e^{-\zeta\omega_n t}$	$\frac{s^2}{s^2+2s\zeta\omega_n+\omega_n^2}$
	$\frac{1}{\omega_n^2}\left[1+\left(\cos\sqrt{1-\zeta^2}\omega_n t+\frac{\zeta}{1-\zeta^2}\sin\sqrt{1-\zeta^2}\omega_n t\right)e^{-\zeta\omega_n t}\right]$	$\frac{1}{s(s^2+2s\zeta\omega_n+\omega_n^2)}$
Third-order systems		
	$\frac{e^{\alpha t}-[1+(\alpha-\beta)t]e^{\beta t}}{(\alpha-\beta)^2}$	$\frac{1}{(s-\alpha)(s-\beta)^2}$
	$\frac{\alpha e^{\alpha t}-[\alpha+\beta(\alpha-\beta)t]e^{\beta t}}{(\alpha-\beta)^2}$	$\frac{s}{(s-\alpha)(s-\beta)^2}$
	$\frac{\alpha^2 e^{\alpha t}-[2\alpha-\beta+\beta(\alpha-\beta)t]\beta e^{\beta t}}{(\alpha-\beta)^2}$	$\frac{s^2}{(s-\alpha)(s-\beta)^2}$
	$\frac{(\beta-\gamma)e^{\alpha t}+(\gamma-\alpha)e^{\beta t}+(\alpha-\beta)e^{\gamma t}}{(\alpha-\beta)(\beta-\gamma)(\gamma-\alpha)}$	$\frac{1}{(s-\alpha)(s-\beta)(s-\gamma)}$

TABLE B.1 *Continued*

	Signal $f(t)$	Laplace transform $F(s)$
Various functions		
	$\dfrac{\alpha\sin\beta t - \beta\sin\alpha t}{\alpha\beta(\alpha^2-\beta^2)}$	$\dfrac{1}{(s^2+\alpha^2)(s^2+\beta^2)}$
	$\dfrac{\cos\beta t - \cos\alpha t}{\alpha^2-\beta^2}$	$\dfrac{s}{(s^2+\alpha^2)(s^2+\beta^2)}$
	$\dfrac{1}{\sqrt{\pi t}}$	$\dfrac{1}{\sqrt{s}}$
	$2\sqrt{\dfrac{t}{\pi}}$	$\dfrac{1}{s\sqrt{s}}$
	$\dfrac{n!}{(2n)!}\dfrac{4^n}{\sqrt{\pi}}t^{n-1/2}$	$\dfrac{1}{s^n\sqrt{s}}$
	$\dfrac{1}{\sqrt{\pi t}}e^{\alpha t}$	$\dfrac{1}{\sqrt{s-\alpha}}$
Error functions		
	$\dfrac{2}{\sqrt{\alpha\pi}}\int_0^{\sqrt{\alpha t}} e^{-\zeta^2}d\zeta$	$\dfrac{1}{s\sqrt{s+\alpha}}$
	$\dfrac{2e^{-\alpha t}}{\sqrt{\pi(\beta-\alpha)}}\int_0^{\sqrt{(\beta-\alpha)t}} e^{-\zeta^2}d\zeta$	$\dfrac{1}{(s+\alpha)\sqrt{s+\beta}}$
	$\dfrac{e^{-\alpha t}}{\sqrt{\pi t}} + 2\sqrt{\dfrac{\alpha}{\pi}}\int_0^{\sqrt{\alpha t}} e^{-\zeta^2}d\zeta$	$\dfrac{\sqrt{s+\alpha}}{s}$
Bessel function of zero order	$I_0(\alpha t)$	$\dfrac{1}{\sqrt{s^2+\alpha^2}}$
Modified Bessel function of order zero	$J_0(\alpha t)$	$\dfrac{1}{\sqrt{s^2-\alpha^2}}$

where

$$F_1(s) = L\{f_1(t)\}$$
$$F_2(s) = L\{f_2(t)\}$$

and χ 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[3,37] 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:

$$L^{-1}\{F_1(s)\cdot F_2(s)\} = \int_0^t f_1(\tau) f_2(t-\tau)d\tau \tag{B.14}$$

The integral on the right-hand side of Eq. (B.14) is called *convolution* and is often written as

$$\int_0^t f_1(\tau) f_2(t-\tau)d\tau = f_1(t) * f_2(t)$$

B.4.4 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 τ (Fig. B.8). What is the Laplace transform of the delayed signal $f(t-\tau)$? The derivation[3,37] yields

$$L\{f(t-\tau)\} = F(s)e^{-s\tau} \tag{B.15}$$

Numerical Example We shall calculate the Laplace transform of a rectangular pulse (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 τ. The Laplace transform of the first unit function is given by $1/s$, and 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,

$$f(t) = \begin{cases} 0 & t < 0 \\ 1 & 0 \le t \le \tau \\ 0 & t > \tau \end{cases}$$

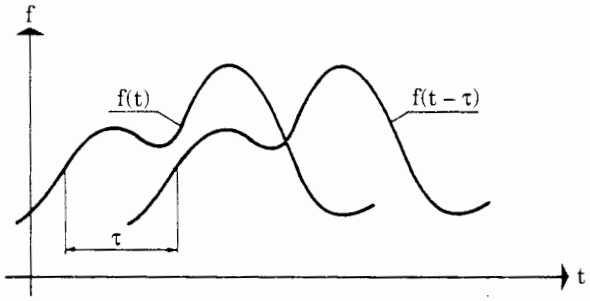


Figure B.8 Plot of a function $f(t)$ displaced by a time delay τ.

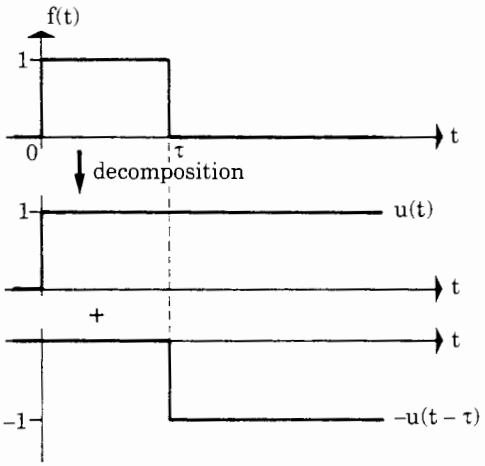


Figure B.9 If the Laplace transform of a single square-wave pulse (top trace) has to be calculated, the pulse is first decomposed into two unit step functions (center and bottom traces).

B.4.5 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 $\int_0^t f(t)\,dt$. The Laplace transform of the derivative is[3,37]

$$L\left\{\frac{df(t)}{dt}\right\} = sF(s) - f(0) \tag{B.16}$$

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[3]

$$L\left\{\int_0^t f(t)dt\right\} = \frac{F(s)}{s} + \frac{f^{(-1)}(0)}{s} \tag{B.17}$$

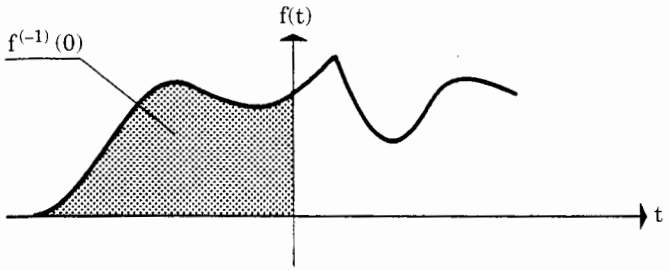


Figure B.10 The term $f^{(-1)}(0)$ is given by the shaded area under the curve $f(t)$.

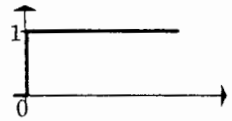
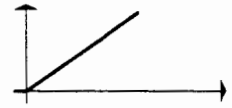
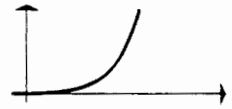
Oscillogram	Type of signal	Original function f(t)	Laplace Transform F(s)
	Step function	$f(t) = u(t)$	$F(s) = \frac{1}{s}$
	Ramp	$f(t) = t \quad (t \geq 0)$	$F(s) = \frac{1}{s^2}$
	Parabola	$f(t) = \frac{t^2}{2} \quad (t \geq 0)$	$F(s) = \frac{1}{s^3}$

Figure B.11 Example of applying the rule of integration. The Laplace transforms of a unit step, a ramp, and a parabola are determined.

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 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). 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 ∞; 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 understand better 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 1 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 unit 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 unit 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.

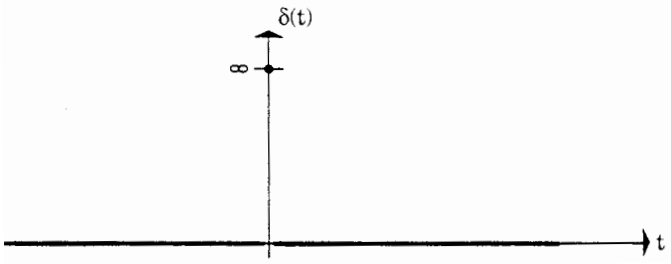


Figure B.12 Plot of the delta function $\delta(t)$.

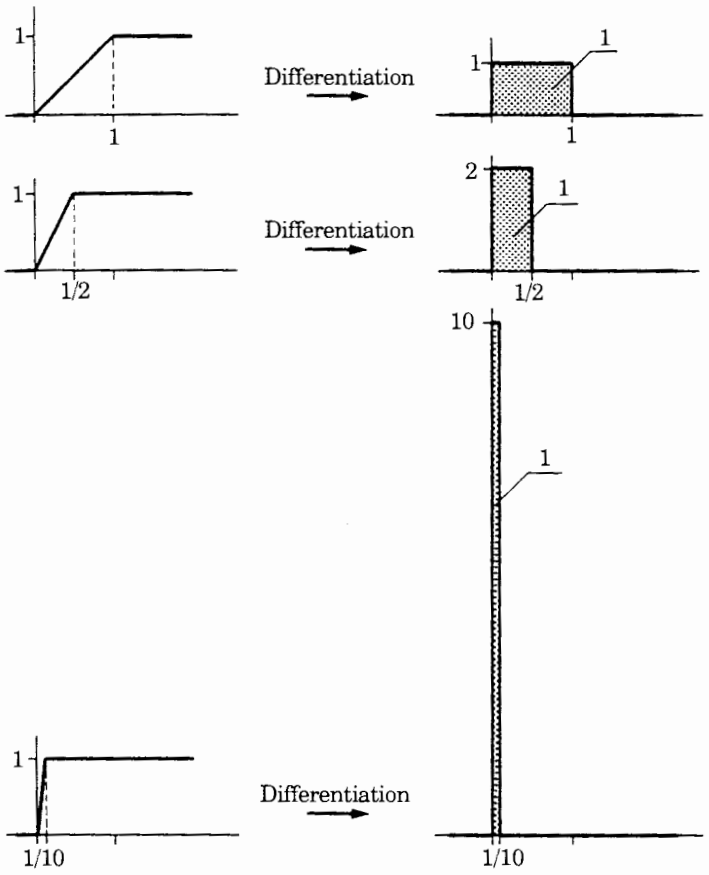


Figure B.13 Practical approximation of the delta function.

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

$$F(s) = 1$$

B.4.6 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.[3] The initial value theorem reads:

$$f(0) = \lim_{s\to\infty} sF(s) \tag{B.18}$$

The final-value theorem reads:

$$f(\infty) = \lim_{s\to 0} sF(s) \tag{B.19}$$

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

$$f(0) = \lim_{s\to\infty} sF(s) = \lim_{s\leftarrow\infty} \frac{s}{s(1+sT)} = 0$$

On the other hand, the final value, according to Eq. (B.19), is

$$f(\infty) = \lim_{s\to 0} sF(s) = \lim_{s\leftarrow 0} \frac{s}{s(1+sT)} = 1$$

B.5 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 $0 \le t \le \infty$. 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.

B.6 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,

$$u(t) = Ri(t) \tag{B.20a}$$

for inductors,

$$u(t) = L\frac{di}{dt} \tag{B.20b}$$

and for capacitors,

$$u(t) = \frac{1}{C}\int_0^t i\,dt \tag{B.20c}$$

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,

$$U(s) = RI(s) \tag{B.21a}$$

for inductors,

$$U(s) = L[sI(s) - i(0)] \tag{B.21b}$$

and for capacitors,

$$U(s) = \frac{1}{C}\left[\frac{I(s)}{s} + \frac{i^{(-1)}(0)}{s}\right] \tag{B.21c}$$

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 $U(s) = sLI(s)$ and the quotient

$$\frac{U(s)}{I(s)} = sL \tag{B.22a}$$

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

$$\frac{U(s)}{I(s)} = \frac{1}{sC} \tag{B.22b}$$

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 us 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 that 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

$$U_1(s) = RI(s) + \frac{1}{sC} I(s)$$
$$U_2(s) = RI(s)$$

After the elimination of $I(s)$ we obtain

$$U_2(s) = \frac{sRC}{1 + sRC} U_1(s)$$

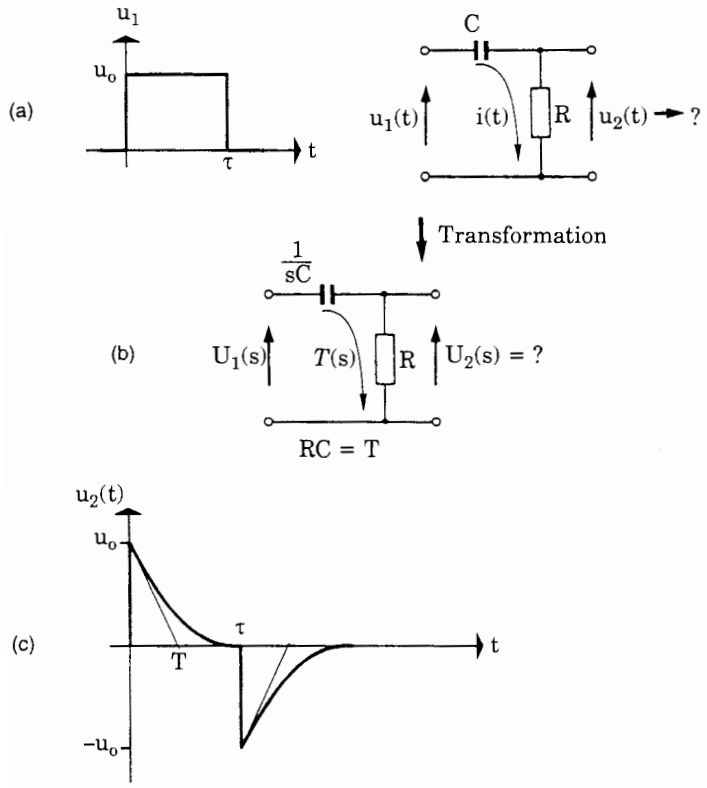


Figure B.14 Calculating the transient response of the passive *RC* differentiator using the Laplace transform. (*a*) Defining the variables in the time domain. (*b*) Introducing variables in the complex-frequency domain. (*c*) Plot of the output signal u_2 against time.

This can be written as

$$U_2(s) = F(s)U_1(s) \tag{B.23}$$

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 τ 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:

$$U_2(s) = u_0 \frac{1}{s + 1/T} - u_0 \frac{e^{-s\tau}}{s + 1/T}$$

From the table of Laplace transforms (Table B.1), the signal corresponding to the first term of Eq. (B.24*b*) is

$$u_{21}(t) = u_0 e^{-(t-\tau)/T} \tag{B.25a}$$

The second term in Eq. (B.24*b*) corresponds to a decaying exponential function delayed by τ,

$$u_{22}(t) = -u_0 e^{-(t-\tau)/T} \tag{B.25b}$$

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.14*c*.

B.7 Closing the Gap between the Time Domain and the Complex Frequency Domain

As demonstrated in the previous section (Eq. B.23), the Laplace transforms of input and output signals of any linear electric network are related by the transfer function $F(s)$,

$$U_2(s) = U_1(s)F(s)$$

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

$$F(s) = L\{h(t)\} \tag{B.26}$$

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

$$F(s) = \frac{U_2(s)}{U_1(s)} = \frac{1}{1+sT}$$

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_0T_0, where $U_0 = 1\,\mathrm{V}$ and $T_0 = 1\,\mathrm{s}$. For simplicity, this factor is omitted here and in the following calculations.

B.8 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

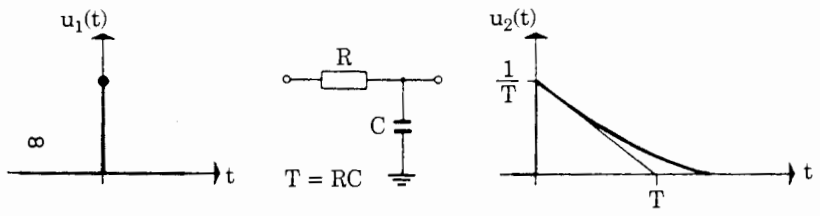


Figure B.15 Calculating the transient response of the passive RC integrator by using the Laplace transform.

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:

$$u_2(t) = i(t)R + u_2(t)$$
$$u_2(t) = \frac{1}{C}\int_0^t i\,dt$$

Applying the Laplace transform to these equations [refer to Eqs. (B.21)], we obtain

$$U_1(s) = I(s)R + U_2(s)$$
$$U_2(s) = \frac{1}{C}\left[\frac{I(s)}{s} + \frac{i^{(-1)}(0)}{s}\right]$$

We can eliminate $I(s)$ from the first of these equations. We then get

$$U_2(s) = \frac{1}{C}\left[\frac{U_1(s) - U_2(s)}{Rs} + \frac{i^{(-1)}(0)}{s}\right]$$

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 $-\infty < t < 0$; that is, $i^{(-1)}(0)$ 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 $u_2(t)$

$$u_2(t) = \frac{1}{T}e^{-t/T} + u(0)e^{-t/T}$$

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_0T_0, where $U_0 = 1\,\text{V}$ and $T_0 = 1\,\text{s}$.]

B.9 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,

$$F(s) = \frac{a_n s^n + a_{n-1}s^{n-1} + \ldots + a_1 s + a_0}{b_m s^m + b_{m-1}s^{m-1} + \ldots + b_1 s + b_0} \qquad m \geq n \tag{B.27a}$$

This transfer function can also be written in the factored form:

$$F(s) = \frac{(s-\alpha_1)(s-\alpha_2)\ldots(s-\alpha_n)}{(s-\beta_1)(s-\beta_2)\ldots(s-\beta_m)} \tag{B.27b}$$

Here the α_i values are the zeros and the β_i values the poles of the transfer function.

The α_i and β_i can be either real or complex. For example, a real value of β_i corresponds to a pole located on the real axis (σ axis) in the complex s plane (refer to Fig. B.16). If a complex pole is located at $\beta_i = A + jB$, another conjugate complex pole will exist at $\beta_i^* = A - jB$, where the asterisk denotes the conjugate value of β_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 zeros of the network transfer function $F(s)$ are known. To transform Eq. (B.27*b*) back into the time domain, it is most convenient to decompose this expression into partial fractions:

$$F(s) = \underbrace{\frac{R_1}{s-\beta_1} + \frac{R_2}{s-\beta_2} + \cdots}_{\substack{\text{partial fractions generated} \\ \text{by single real poles}}} + \underbrace{\frac{R_i}{s-(A_i+jB_i)} + \frac{R_i^*}{s-(A_i-jB_i)} + \cdots}_{\substack{\text{2 partial fractions generated by} \\ \text{one pair of conjugate complex poles}}} \tag{B.27c}$$

The terms $R_1, R_2, \ldots$ are constants and are called *residues*.[3,37] In Eq. (B.27*c*) 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

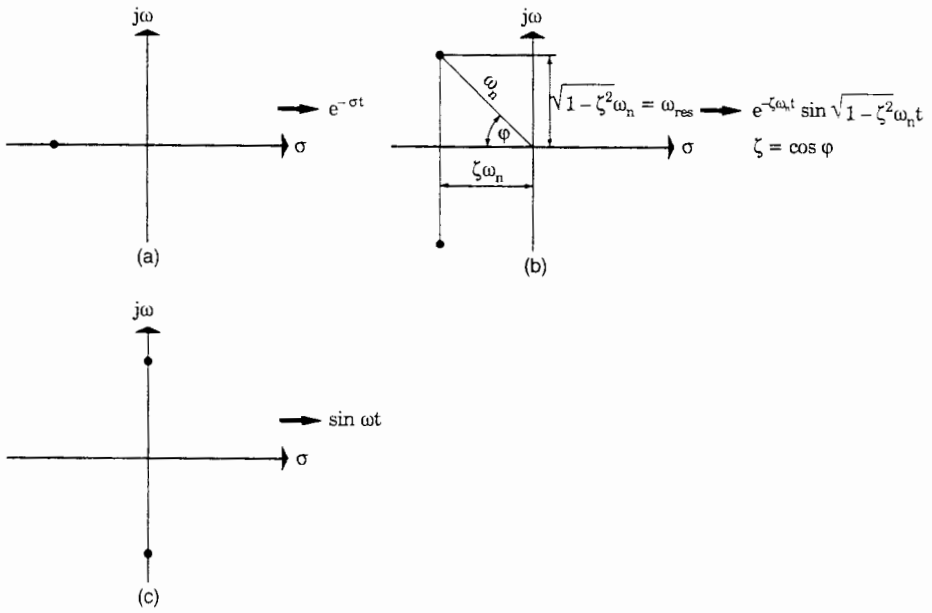


Figure B.16 Calculating the transient response of first- and second-order systems from the pole positions. (*a*) First-order system, pole on the negative σ-axis. (*b*) Second-order system, complex conjugate pole pair in the negative half-plane (damped oscillation). (*c*) Second-order system, poles on the imaginary axis (undamped oscillation).

$$F(s) = \frac{R_i}{s - \alpha_i}$$

is

$$F(t) = R_i \exp(\alpha_i t)$$

Hence the contribution of the single real poles to the signal $f(t)$ is

$$f(t)_{\text{single poles}} = R_1 \exp(\alpha_1 t) + R_2 \exp(\alpha_2 T) + \ldots \tag{B.28}$$

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:

$$F(s)_{\text{pole pair}} = \frac{R_i}{s - (A + jB_i)} + \frac{R_i{}^*}{s - (A - jB_i)} \tag{B.29a}$$

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; i.e., in the most general case we have

$$R_i = a + jb$$
$$R_i^* = a - jb$$

where a and b are real constants.

This is easily proved by combining the two terms in Eq. (B.29a) over a common denominator:

$$F(s)_{\text{pole pair}} = \frac{s(R_i + R_i^*) - A(R_i + R_i^*) + jB(R_i - R_i^*)}{s^2 - 2As + (A^2 + B^2)} s \tag{B.29b}$$

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 above. Equation (B.29a) can therefore be rewritten as

$$F(s)_{\text{pole pair}} = \frac{a + jb}{s - (A + jB)} + \frac{a - jb}{s - (A - jB)} \tag{B.29c}$$

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 α. Applying this to Eq. (B.29c), we obtain

$$f(t)_{\text{pole pair}} = a[e^{At}e^{jBt} + e^{At}e^{-jBt}] + jb[e^{At}e^{jBt} - e^{At}e^{-jBt}] \tag{B.29d}$$

Using the well-known Euler theorem,

$$\sin x = \frac{e^{jx} - e^{-jx}}{2j}$$
$$\cos x = \frac{e^{jx} + e^{-jx}}{2}$$

we can write

$$f(t)_{\text{pole pair}} = 2ae^{At}\cos Bt - 2be^{At}\sin Bt \tag{B.29e}$$

This can be brought into the more general form

$$f(t)_{\text{pole pair}} = Ce^{At}\cos(Bt + \Phi) \tag{B.29f}$$

where

$$C = 2\sqrt{a^2 + b^2}$$
$$\Phi = -\tan^{-1}\frac{b}{a}$$

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.

B.10 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 σ axis ($\alpha < 0$) gives rise to the decaying exponential function (see Table B.2)

$$f(t) \propto e^{\alpha t} \qquad \alpha < 0$$

This is an exponential function having a real exponent; hence α is called a *real frequency*.

A complex pole pair located on the imaginary axis [$A = 0$ in Eq. (B.29*f*)] 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 proved 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.29*c*)]:

$$\begin{aligned} F(s)_{\text{pole pair}} &= \frac{a+jb}{s-(A+jB)} + \frac{a-jb}{s-(A-jB)} \\ &\rightarrow \frac{a+jb}{s-(-\zeta\omega_n + j\sqrt{1-\zeta^2}\,\omega_n)} \\ &\quad + \frac{a-jb}{s-(-\zeta\omega_n - j\sqrt{1-\zeta^2}\,\omega_n)} \end{aligned} \tag{B.29g}$$

TABLE B.2 Laplace Transforms of Generalized First- and Second-Order Systems

$F(s)$	$f(t)$
$\dfrac{R_i}{s-\alpha_i}$	$R_i e^{\alpha t}$
$\dfrac{R_i}{s-(A+jB)} + \dfrac{R_i^*}{s-(A-jB)}$	$Ce^{At}\cos(Bt+\Phi)$

Comparing the coefficients on both sides of the arrow, we obtain the equalities

$$\begin{aligned} A &= -\zeta\omega_n \\ B &= \sqrt{1-\zeta^2}\,\omega_n \end{aligned} \tag{B.30}$$

where ζ and ω_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

$$f(t) = \exp(-\zeta\omega_n t)\cos(\sqrt{1-\zeta^2}\,\omega_n t + \Phi) \tag{B.31}$$

The time constant of the decaying exponential function in Eq. (B.31) is given by $1/(\zeta\omega_n)$; the frequency of the damped oscillation is given by $\omega_{res} = \sqrt{1-\zeta^2}\,\omega_n$. 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 ω_n, as shown in Fig. B.16*b*.

Appendix

C

Digital Filter Basics

In the age of all-digital PLLs and software PLLs, increasing use of digital filters is made. This appendix is a short overview on digital filter design.

C.1 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 $H(j\omega)$, but can also be written $H(\omega)$.] $H(s)$ is defined to be

$$H(s) = \frac{O(s)}{I(s)} \tag{C.1}$$

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

$$O(s) = I(s)H(s) = H(s) \tag{C.2}$$

i.e., 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.1*a* 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

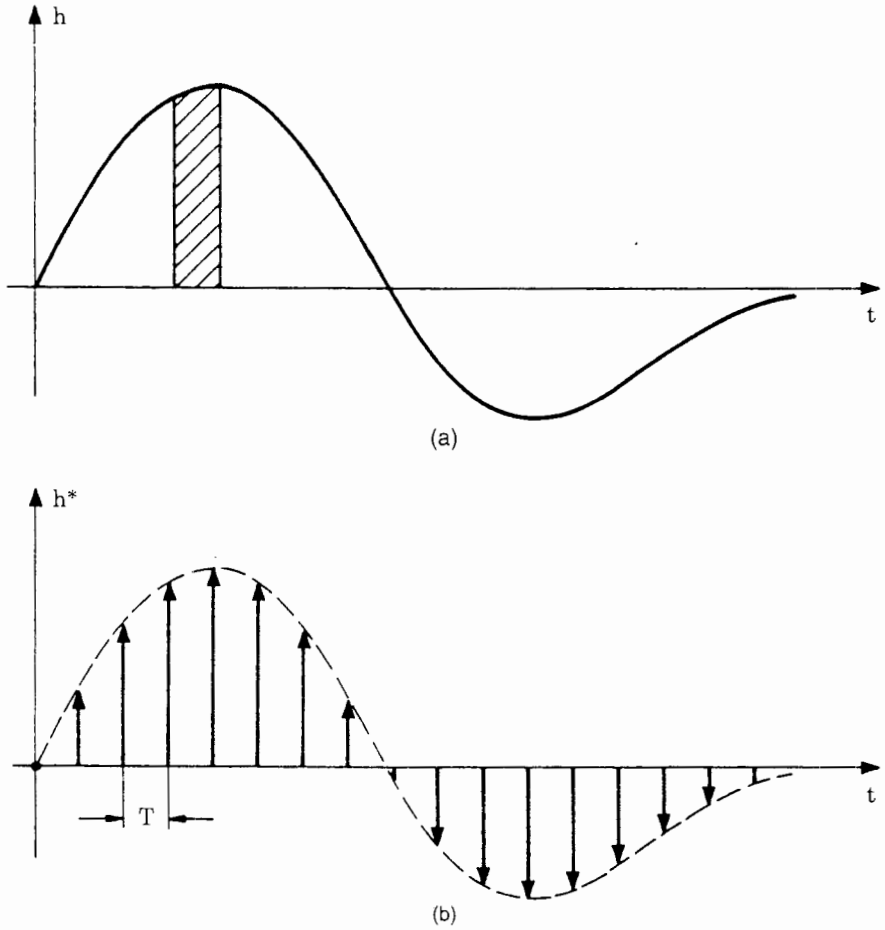


Figure C.1 Impulse responses of analog and digital filters. (*a*) Impulse response $h(t)$ of an analog filter. (*b*) Impulse response $h^*(t)$ of a digital filter. $h^*(t)$ is a sampled version of $h(t)$.

infinite, because $h(t)$ asymptotically approaches zero (or a constant value) for $t \to \infty$. 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 by Fig. C.1*b*. 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.1*b* 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.1*b* we can write

$$h^*(t) = T\sum_{n=0}^{\infty} h(nT)\delta(t - nT) \tag{C.3}$$

where $h(nT)$ is the amplitude of the nth 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.1*a*) in the interval $nT \le t < (n + 1)T$. 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; i.e., we write $h(n)$ for $h(nT)$, etc.

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

$$H^*(s) = T\sum_{n=0}^{\infty} h(n)e^{-snT} \tag{C.4}$$

We note that the complex frequency s appears only in the form of e^{-snT}. To simplify the notation, we introduce the substitution

$$z = e^{sT} \tag{C.5}$$

This is the definition of the *z transform*. Using Eq. (C.5), we can rewrite the transfer function of the digital filter as

$$H(z) = H^*(s)|_{z=e^{sT}} = T\sum_{n=0}^{\infty} h(n)z^{-n} \tag{C.6}$$

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 the *z operator* and 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 exists only at the sampling instants $t = 0, T, 2T, \ldots, nT$, 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 $x(t - 2T)$, etc. As we mentioned above, 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 *infinite impulse response* (IIR) 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 *finite impulse response* (FIR) filters. Though FIR filters look simpler at a first glance, the corresponding theory is more complex, so we start the discussion with the IIR filter.

C.2 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 survived, however: (1) the impulse-invariant z transform and (2) the bilinear z transform. The second method is by far more important, but it is easier to start with the first.

C.2.1 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.2*a* shows the continuous function $h(t)$ again, Fig. C.2*b* the sampling function $w(t)$, and Fig. C.2*c* the result of the multiplication. Obviously, the transfer function $H^*(s)$ of the IIR filter is obtained by transforming the impulse response $h^*(t) = h(t)w(t)T$ 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; i.e., we have

$$H^*(s) = \frac{T}{2\pi j} H(s) * W(s) = \frac{T}{2\pi j} \oint_C H(\chi) W(s-\chi) d\chi \tag{C.7}$$

In this integral, χ is an auxiliary variable for complex frequency and C is the contour along which the integration has to 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

$$W(s) = 1 + e^{-sT} + e^{-2sT} + \cdots \tag{C.8a}$$

This is clearly a geometric series, and using the formula for the sum of geometric series leads to

$$W(s) = \frac{1}{1 - e^{-sT}} \tag{C.8b}$$

When Eq. (C.8*b*) is inserted into Eq. (C.7), $H^*(s)$ becomes

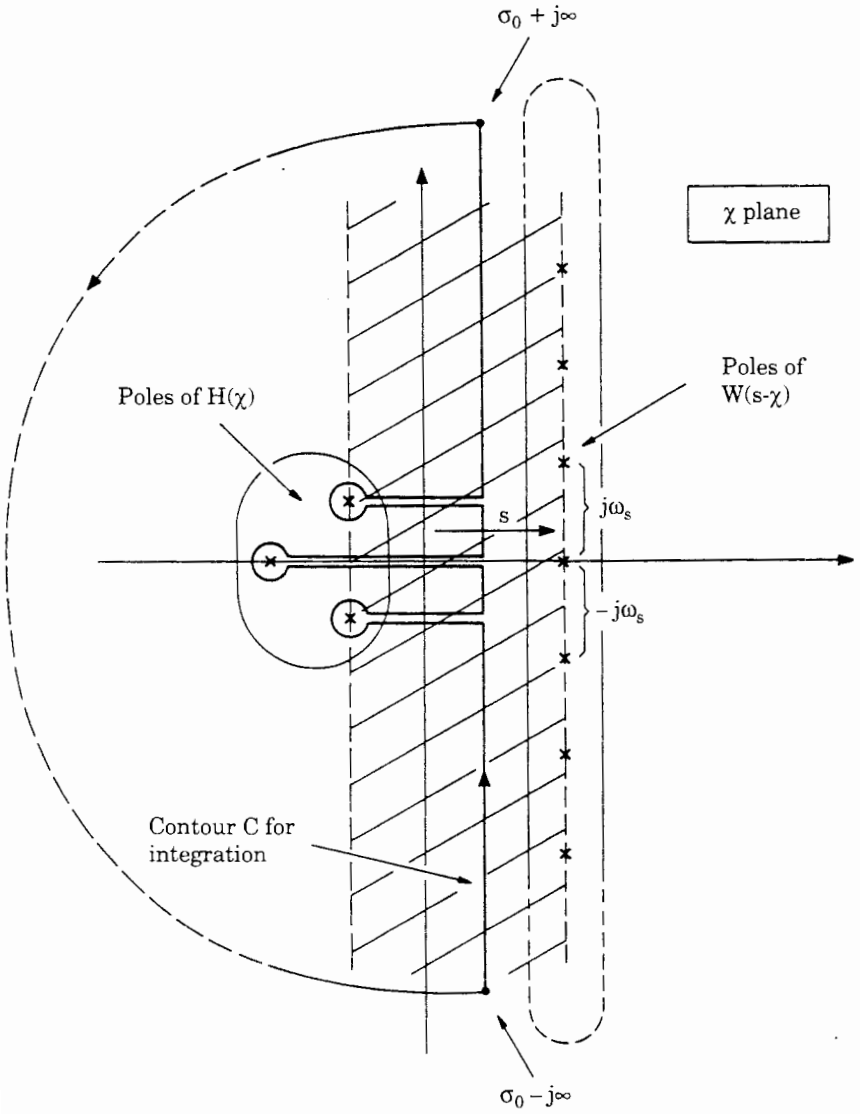


Figure C.3 Locations of the poles of $H(\chi)$ and $W(s - \chi)$ in the χ-plane. The closed curve is the integration path for the convolution integral.

system, they must be in the left half of the χ 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 $-\infty < k < \infty$.[37] 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),[37] and we get

$$H^*(s) = T \sum_{\substack{\text{poles of} \\ H(\chi)}} \frac{\text{Res } H(\chi)}{1 - e^{-(s-\chi)T}} \tag{C.10a}$$

In Eq. (C.10a), Res $H(\chi)$ denotes the residue of the function $H(\chi)$ at $\chi = \chi_0$, where χ_0 is the location of a pole of $H(\chi)$. Since $H^*(s)$ is the transfer function of a sampled system, the s operator appears only in the form of e^{sT}, so we can again substitute z for e^{sT} and obtain

$$H(z) = H^*(s)|_{z=e^{sT}} = T \sum_{\substack{\text{poles of} \\ H(\chi)}} \frac{\text{Res } H(\chi)}{1 - z^{-1}e^{\chi T}} \tag{C.10b}$$

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 $H(s)$; i.e., the impulse-invariant IIR filter has the same number of poles as the corresponding analog filter.

TABLE C.1 Table of the Impulse-Invariant z Transform for Filters of Order 0 to 2

Filter type	$H(s)$	$H(z)$
Zero-order term	1	1
Integrator	$\dfrac{1}{sT_i}$	$\dfrac{T}{T_i}\dfrac{1}{1-z^{-1}}$
First-order lag	$\dfrac{1}{1+s/\omega_0}$	$\omega_0 T \dfrac{1}{1-z^{-1}\exp(-\omega_0 T)}$
Second-order lag	$\dfrac{1}{1+2\zeta(s/\omega_0)+(s^2/\omega_0^2)}$	$\dfrac{z^{-1}(\omega_0 T/\sqrt{1-\zeta^2})\exp(-\zeta\omega_0 T)\sin(\omega_0\sqrt{1-\zeta^2}\,T)}{1-z^{-1}2\exp(-\zeta\omega_0 T)\cos(\omega_0\sqrt{1-\zeta^2}\,T)+z^{-2}\exp(-2\zeta\omega_0 T)}$
Second-order lag, linear term in numerator	$\dfrac{s/\omega_1}{1+2\zeta(s/\omega_0)+(s^2/\omega_0^2)}$	$\dfrac{\omega_0^2 T}{\omega_1}\dfrac{1-z^{-1}\exp(-\zeta\omega_0 T)[\cos(\omega_0\sqrt{1-\zeta^2}\,T)+(\zeta/\sqrt{1-\zeta^2})\sin(\omega_0\sqrt{1-\zeta^2}\,T)]}{1-z^{-1}2\exp(-\zeta\omega_0 T)\cos(\omega_0\sqrt{1-\zeta^2}\,T)+z^{-2}\exp(-2\zeta\omega_0 T)}$
Second-order lag, constant + linear term in numerator	$\dfrac{1+s/\omega_1}{1+2\zeta(s/\omega_0)+(s^2/\omega_0^2)}$	$\dfrac{\omega_0^2 T}{\omega_1}\dfrac{1-z^{-1}\exp(-\zeta\omega_0 T)\{\cos(\omega_0\sqrt{1-\zeta^2}\,T)+[(\omega_1/\omega_0-\zeta/\sqrt{1-\zeta^2}]\sin(\omega_0\sqrt{1-\zeta^2}\,T)\}}{1-z^{-1}2\exp(-\zeta\omega_0 T)\cos(\omega_0\sqrt{1-\zeta^2}\,T)+z^{-2}\exp(-2\zeta\omega_0 T)}$

It follows from Eq. (C.10*b*) that the z-transfer function can always be represented as the ratio of two polynomials in z, whose order is finite; i.e., we have

$$H(z) = \frac{O(z)}{I(z)} = \frac{b_0 + b_1 z^{-1} + \cdots + b_M z^{-M}}{1 + a_1 z^{-1} + \cdots + a_N z^{-N}} \tag{C.10c}$$

The a_i and b_i ($i = 0, 1, 2, \ldots$) are called *filter coefficients*. There is no restriction on the size of the orders N and M of the polynomials in Eq. (C.10*c*). M can be smaller, equal to, or larger than N. Taking all terms containing $O(z)$ to the left side of the equation gives

$$O(z) + a_1 z^{-1} O(z) + \cdots = b_0 I(z) + b_1 z^{-1} I(z) + \cdots \tag{C.10d}$$

If this is transformed back into the time domain and the sampling interval T is dropped everywhere, we get

$$o(n) = b_0 i(n) + b_1 i(n-1) + \cdots - a_1 o(n-1) - a_2 o(n-2) - \cdots \tag{C.10e}$$

which is a recursion for the output signal at sampling instant $t = nT$. The IIR filter is therefore also referred to as *recursive filter*. Equation (C.10*e*) says that $o(n)$ is calculated from a weighted sum of one or several input signals $i(n)$, $i(n-1), \ldots$ 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

$$H^*(j\omega) = \frac{T}{2\pi} \oint_C H(j\eta) W[j(\omega - \eta)] d\eta \tag{C.11}$$

where η 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.4*a*, and its spectrum in Fig. C.4*b*. Let us investigate the result of the convolution by the example of Fig. C.5. Figure C.5*a* shows the frequency response $H(j\omega)$ of a low-pass filter. Figure C.5*b* 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 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 $\omega_n = \omega_s/2$; ω_n is called the *Nyquist frequency*. The result of the convolution is shown in Fig. C.5*c*. 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

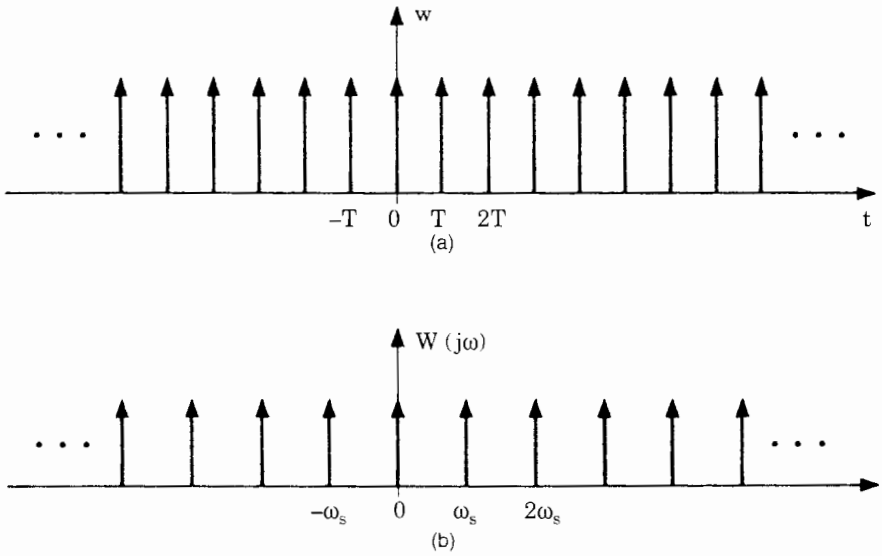


Figure C.4 Sampling function $w(t)$ and its spectrum $W(j\omega)$. (a) Sampling function $w(t)$. (b) Spectrum $W(j\omega)$.

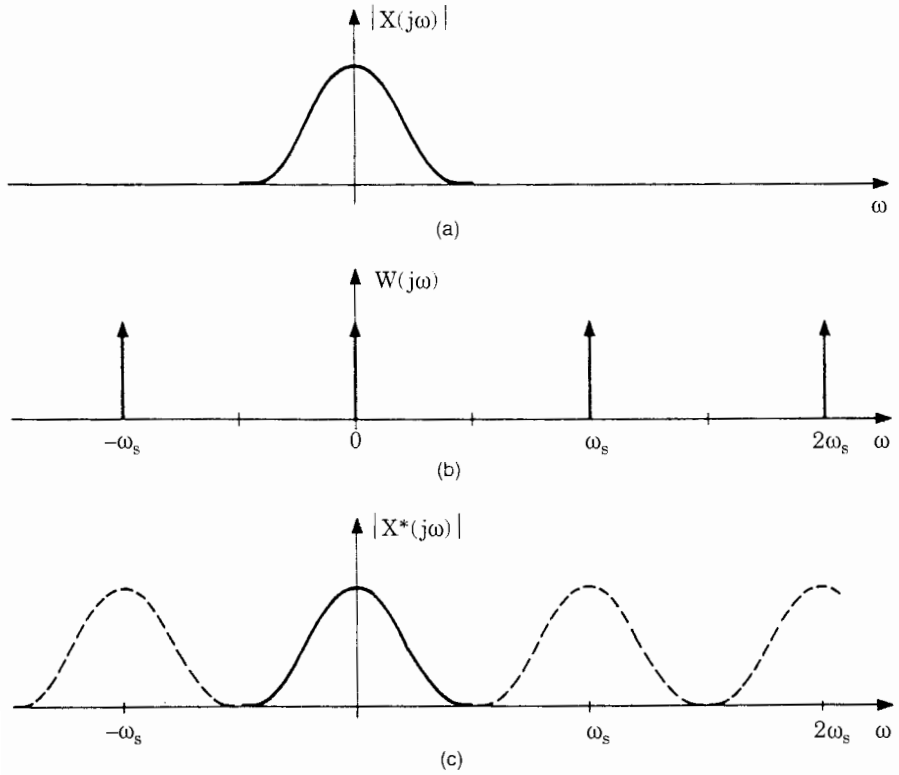


Figure C.5 Result of the convolution of $X(j\omega)$ and $W(j\omega)$ for the case of negligible aliasing. (a) Frequency response $X(j\omega)$ of an analog filter. (b) Spectrum $W(j\omega)$ of the sampling function $w(t)$. (c) Result of the convolution $X(j\omega)*W(j\omega)$.

the angular sampling frequency ω_s. The solid curve in Fig. C.5*c* is the so-called *main lobe*; 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 are going to consider now an example where the gain $H(j\omega)$ has a value much greater than 0 at the Nyquist frequency (Fig. C.6*a*). Figure C.6*b* once more shows the spectrum $W(j\omega)$ of the sampling function $w(t)$, and Fig. C.6*c* 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) dictates that the sampling frequency must be chosen at least twice a critical frequency ω_c, where the gain $H(j\omega_c)$ is so low that it can be neglected. A numerical example will demonstrate the implications of the sampling theorem.

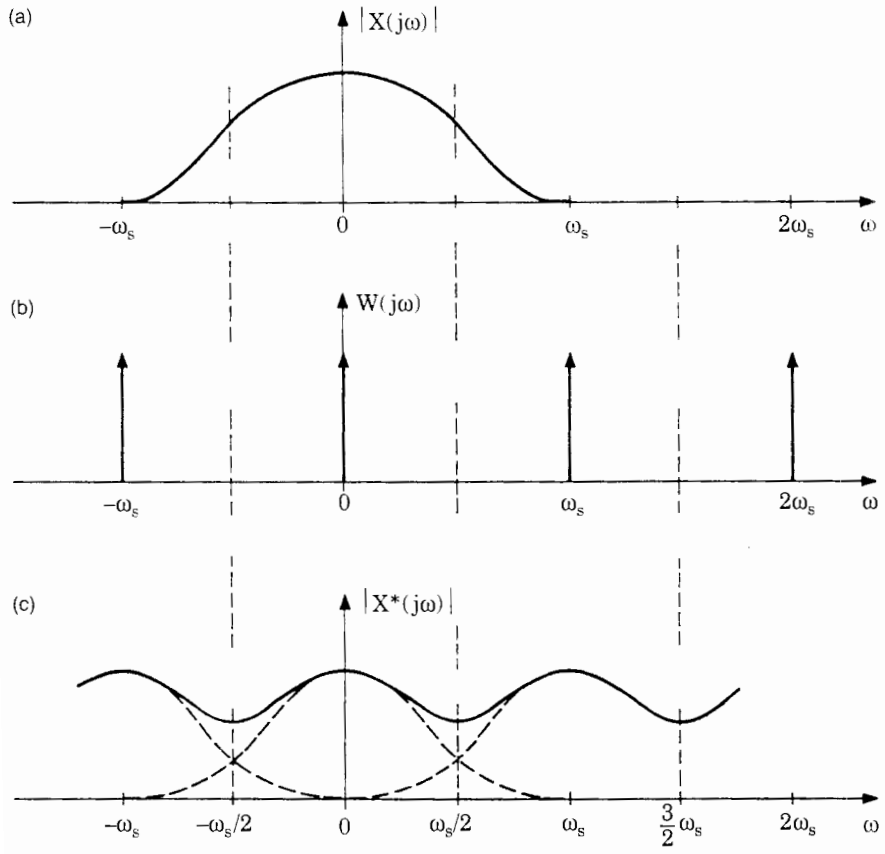


Figure C.6 Conditions are similar to those in Fig. C.5, but with severe aliasing. The gain $X(j\omega)$ of the analog filter is too high at the Nyquist frequency $\omega_n = \omega_s/2$. (*a*) Frequency response $X(j\omega)$ of an analog filter. (*b*) Spectrum $W(j\omega)$ of the sampling function $w(t)$. (*c*) Result of the convolution $X(j\omega)^*W(j\omega)$.

Numerical Example Assume that we want to design a digital two-pole low-pass filter having a 3-dB corner frequency of 1 kHz. A corresponding analog low-pass 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, etc. 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,* i.e., 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.

C.2.2 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.7*a* shows the frequency response of an

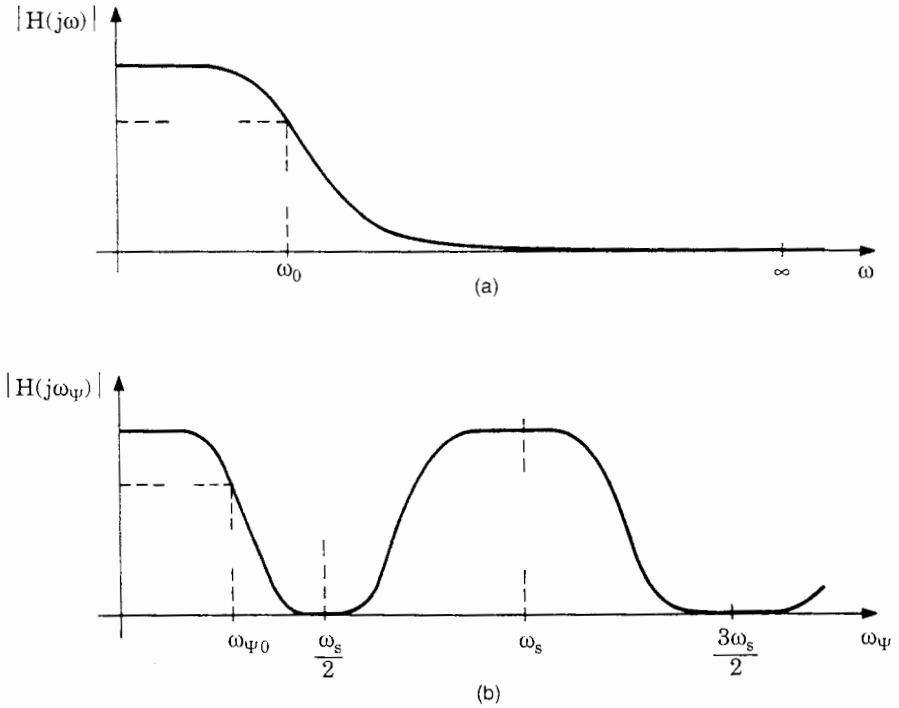


Figure C.7 Principle of the bilinear z transform. (*a*) Frequency response $H(j\omega)$ of an analog filter. (*b*) The frequency axis of (*a*) is projected onto the pseudo-frequency axis ω_Ψ. Transforming the frequency response of the filter in (*a*) into the pseudo-frequency domain results in the periodic frequency response $H^*(j\omega)$.

analog filter; again a low-pass filter has been chosen. Its 3-dB (angular) corner frequency is denoted ω_0. We now introduce a new radian frequency variable, the so-called pseudo-frequency ω_ψ. We define that the pseudo-frequency ω_ψ and the (radian) frequency ω are related by a tangent transform

$$\omega = \frac{2}{T}\tan\frac{\omega_\psi T}{2} \tag{C.12}$$

i.e., the frequency range $-\infty < \omega < \infty$ is projected onto the pseudo-frequency interval $-\omega_s/2 \le \omega_\psi \le \omega_s/2$, 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, i.e., onto the pseudo-frequency intervals $\omega_s/2 \le \omega_\psi \le 3\omega_s/2$, $3\omega_s/2 \le \omega_\psi \le 5\omega_s/2$, etc. If we transform the frequency response of the analog filter into the pseudo-frequency domain, we get the periodic transfer function $H^*(j\omega)$ as plotted in Fig. C.7*b*. Between $H(j\omega)$ and $H^*(j\omega)$ there is the relation

$$H^*(j\omega_\psi) = H(j\omega)_{\omega=(2/T)\tan(\omega_\psi T/2)} \tag{C.13}$$

which says that the gain of the new filter at pseudo-frequency ω_ψ is identical with the gain of the analog filter at frequency ω, where ω and ω_ψ 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 $\omega_{\psi 0}$ in Fig. C.7*b*, 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 should have the same corner frequency as the original analog filter; i.e., the pseudo corner frequency should be ω_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 ω_{0p}, we have

$$\omega_{0p} = \frac{2}{T}\tan\frac{\omega_0 T}{2} \tag{C.14}$$

It is easily seen then that the pseudo corner frequency becomes ω_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 use now the transformation

$$s = \frac{2}{T}\tanh\frac{s_\psi T}{2} \tag{C.15}$$

When we set $s = j\omega$ and $s_\psi = j\omega_\psi$ in Eq. (C.15), Eq. (C.12) is obtained again. From Eq. (C.15), the transfer function $H^*(s_\psi)$ of the digital filter is calculated from

$$H^*(s_\psi) = H(s)|_{s=(2/T)\tanh(s_\psi T/2)} \tag{C.16}$$

Using Euler's relations, the tanh function can be written in the form of exponential functions:

$$\tanh\frac{s_\psi T}{2} = \frac{1 - e^{-s_\psi T}}{1 + e^{-s_\psi T}} \tag{C.17}$$

Equation (C.16) can therefore be rewritten as

$$H^*(s_\psi) = H(s)|_{s=(2/T)(1-e^{-s_\psi T})/(1+e^{-s_\psi T})} \tag{C.18a}$$

Because the complex pseudo-frequency appears exclusively in the form $\exp(ns_\psi T)$, $\exp(s_\psi T)$ can be replaced by z:

$$z = e^{s_\psi T} \tag{C.18b}$$

i.e., we apply the z transform to the pseudo-frequency domain. Using the substitution of Eq. (C.18*b*), we finally get

$$H(z) = H(s)|_{s=(2/T)(1-z^{-1})/(1+z^{-1})} \tag{C.19}$$

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.10*e*)].

To see the real benefit of the bilinear z transform, let us have a look again at Fig. C.7. The filter in this example is a low-pass filter. The gain $H(j\omega)$ of the analog filter (Fig. C.7*a*) approaches 0 only for $\omega \to \infty$. Because of the tangent transformation, however, the gain of the digital filter becomes *exactly* 0 at the Nyquist pseudo-frequency $\omega_c = \omega_s/2$, as shown in Fig. C.7*b*. Consequently, the

TABLE C.2 Table of the Bilinear z Transform for Filters of Order 0 to 2

Filter type	$H(s)$	$H(z)$
Zero-order term	1	1
Integrator	$\dfrac{1}{sT_i}$	$\dfrac{T}{2T_i}\dfrac{1+z^{-1}}{1-z^{-1}}$
First-order lag	$\dfrac{1}{1+s/\omega_0}$	$\dfrac{1+z^{-1}}{1+2/(\omega_0 T)+z^{-1}[1-2/(\omega_0 T)]}$
Second-order lag	$\dfrac{1}{1+2\zeta(s/\omega_0)+(s^2/\omega_0^2)}$	$\dfrac{1+2z^{-1}+z^{-2}}{1+4\zeta/(\omega_0 T)+4(\omega_0 T)^2+z^{-1}\left[2-8/(\omega_0 T)^2\right]+z^{-2}\left[1+4\zeta/(\omega_0 T)+4(\omega_0 T)^2\right]}$
Second-order lag, linear term in numerator	$\dfrac{s/\omega_1}{1+2\zeta(s/\omega_0)+(s^2/\omega_0^2)}$	$\dfrac{2}{\omega_1 T}\dfrac{1-z^{-2}}{1+4\zeta/(\omega_0 T)+4(\omega_0 T)^2+z^{-1}\left[2-8/(\omega_0 T)^2\right]+z^{-2}\left[1+4\zeta/(\omega_0 T)+4(\omega_0 T)^2\right]}$
Second-order lag, constant plus linear term in numerator	$\dfrac{1+s/\omega_1}{1+2\zeta(s/\omega_0)+(s^2/\omega_0^2)}$	$\dfrac{(1+2/\omega_1 T)+2z^{-1}+z^{-2}(1-2/\omega_1 T)}{1+4\zeta/(\omega_0 T)+4(\omega_0 T)^2+z^{-1}\left[2-8/(\omega_0 T)^2\right]+z^{-2}\left[1+4\zeta/(\omega_0 T)+4(\omega_0 T)^2\right]}$

main and side lobes of the pseudo-frequency response $H^*(j\omega)$ *do not overlap*, which means that this digital filter *does not exhibit aliasing*. This enables us to choose the sampling frequency much lower than with the impulse-invariant IIR filter. The lack of aliasing is specially 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 are going to design a single-pole Butterworth low-pass digital filter. The 3-dB corner frequency of the IIR filter is required to be $f_c = 200\,\text{Hz}$. The transfer function of the corresponding analog filter is given by

$$H(s) = \frac{1}{1+s/\omega_0} \tag{C.20a}$$

The sampling frequency is chosen $f_s = 1000\,\text{Hz}$, 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

$$H(z) = \frac{1 + z^{-1}}{1 + 2/(\omega_{0p}T) + z^{-1}[1 - 2/(\omega_{0p}T)]} \tag{C.20b}$$

Entering numerical values and normalizing the constant term in the denominator to 1, we get

$$H(z) = \frac{0.4208 + 0.4208\,z^{-1}}{1 - 0.1584\,z^{-1}}$$

Transforming back into 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 zeros. The analog filter from which the design starts then will have a transfer function of the form

$$H(s) = \frac{(1 + s/\omega_{01})\left(1 + 2s\zeta_3/\omega_{03} + [s/\omega_{03}]^2\right)(\cdots)}{(1 + s/\omega_{02})\left(1 + 2s\zeta_2/\omega_{04} + [s/\omega_{04}]^2\right)(\cdots)}$$

Elliptic filters, for example, have transfer functions of this kind.[13] 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 parentheses in the numerator. In a similar way, the transfer function may have a real-axis pole (first term in denominator) and one or several complex conjugate pole pairs. The corresponding corner frequencies of this filter would be ω_{01}, ω_{03}, . . . , ω_{02}, ω_{04}, . . . , etc. When an IIR filter having this transfer function has to be designed, all corner frequencies in the numerator and in the denominator have to be prewarped by 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 that are available now.[25,35]

C.3 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 us start again by plotting the impulse response $h(t)$ of an existing analog filter as shown in Fig. C.1*a*. The digital filter is assumed to have an impulse response as shown in Fig. C.1*b*, 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

$$h^*(t) = T\sum_{n=0}^{N-1} h(n)\delta(t - nT) \tag{C.21}$$

The z-transfer function of the FIR filter then becomes [Eq. (C.6)]

$$H(z) = T\sum_{n=0}^{N-1} h(n)z^{-n} \tag{C.22}$$

Transforming back into the time domain, the output signal of the FIR filter at sampling instant $t = nT$ is obtained from

$$o(n) = T\sum_{k=0}^{N-1} i(n)h(n-k) \tag{C.23}$$

This is not a recursion, since $o(n)$ does not depend on previously calculated samples $o(n-1)$, $o(n-2)$, etc. 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 to 1 for simplicity. Because the FIR filter looks like a tapped delay line, the filter length N is also called *number of taps*. The filter coefficients $h(0), h(1), \ldots, h(N-1)$ are nothing more than the values of the impulse response $h^*(t)$ at sampling times $t = 0, T, 2T, \ldots$. For the FIR filter, the coefficients $h(n)$, $n = 0 \ldots N-1$ 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 that do not have an analog counterpart.

Let us first consider two kinds of impulse response that are of particular interest. Figure C.9*a* shows a finite impulse response that is an even function of time t; in Fig. C.9*b* another impulse response is plotted, one that is an odd function of time. From the theory of the Fourier transform it follows that the spectrum

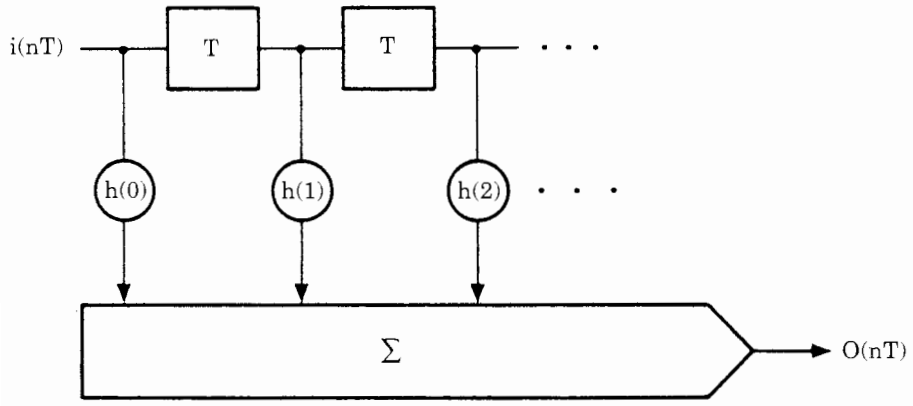


Figure C.8 Signal flow diagram of the FIR filter. The blocks marked with T are delay blocks.

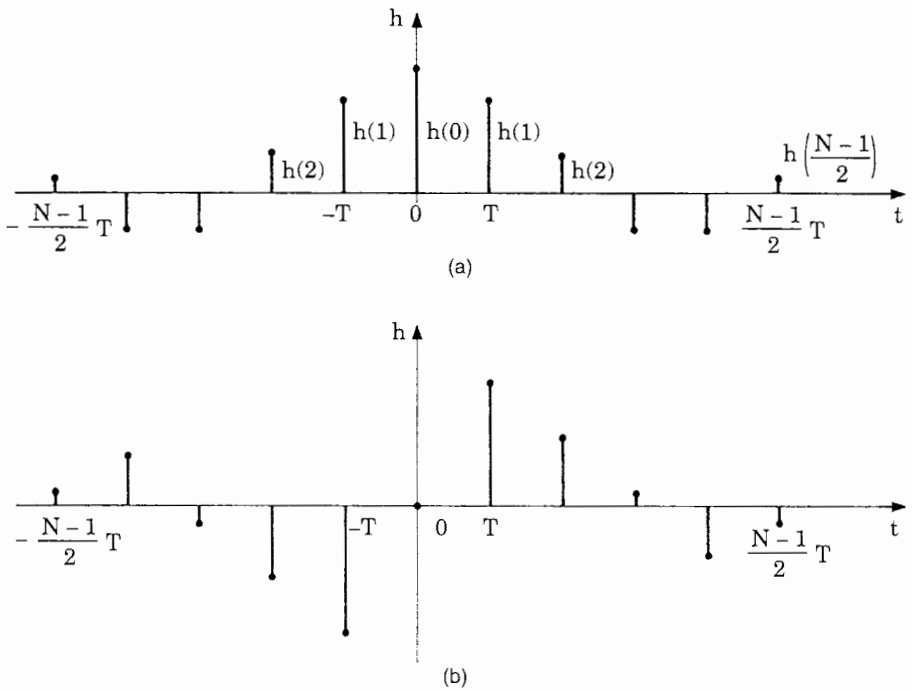


Figure C.9 Two special cases of impulse response of an FIR filter. (*a*) The impulse response is an even function of time. (*b*) The impulse response is an odd function of time.

of a signal that is an even function of time is purely real.[12,13,14] The spectrum of a signal, however, that 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 that has an odd impulse response would be purely imaginary. Its phase shift would be 90° 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 low-pass filter would have a real frequency response $H(j\omega)$ that is exactly 1 within the passband and 0 at all other frequencies. The impulse response $h(t)$ of the ideal low-pass filter would be a sinc function

$$h(t) = \frac{\sin \omega_0 t}{\pi t} \tag{C.24}$$

where ω_0 is the angular corner frequency of the ideal low-pass filter. [Note: *sinc* is an abbreviation for $\sin(\pi x)/(\pi x)$.] The frequency response of the ideal low-pass filter is shown in Fig. C.10*a*, and the impulse response in Fig. C.10*b*. 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.10*c*—is used to cut out a finite portion of the impulse response; i.e., the finite impulse response h^*—as shown in Fig. C.10*d*—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 $h^*(t)$; see also Eq. (C.3).] The impulse response shown in Fig. C.10*d* is finite. If the filter were realizable, its frequency response could be made to approximate very closely that of the ideal low-pass filter by choosing a large number of samples N. Let $H_{\text{zph}}(j\omega)$ be the frequency response of this (unrealizable) filter (Fig. C10*e*). To get a realizable filter, a second modification must be made. The impulse response of Fig. C.10*d* is delayed such that it starts at $t = 0$. If the required time delay is τ, the frequency response of the modified FIR filter is given by

$$H_{\text{lph}}(j\omega) = H_{\text{zph}}(j\omega)e^{-j\omega\tau} \tag{C.25}$$

The phase of term $H_{\text{zph}}(j\omega)$ is zero by definition. The phase of the second term in Eq. (C.25) varies linearly with frequency; i.e., we have

$$\Phi(\omega) = -\omega\tau \tag{C.26}$$

Consequently, the obtained FIR filter has linear phase, and its frequency response is denoted $H_{\text{lph}}(j\omega)$ 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 that have occurred prior to t and from a number of samples that 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; i.e., its impulse response is derived from the impulse response of the fictive ideal filter. When the Parks-McClellan algorithm is used, however, the filter is designed directly in the frequency domain. The term *frequency sampling* was formerly used for this approach.

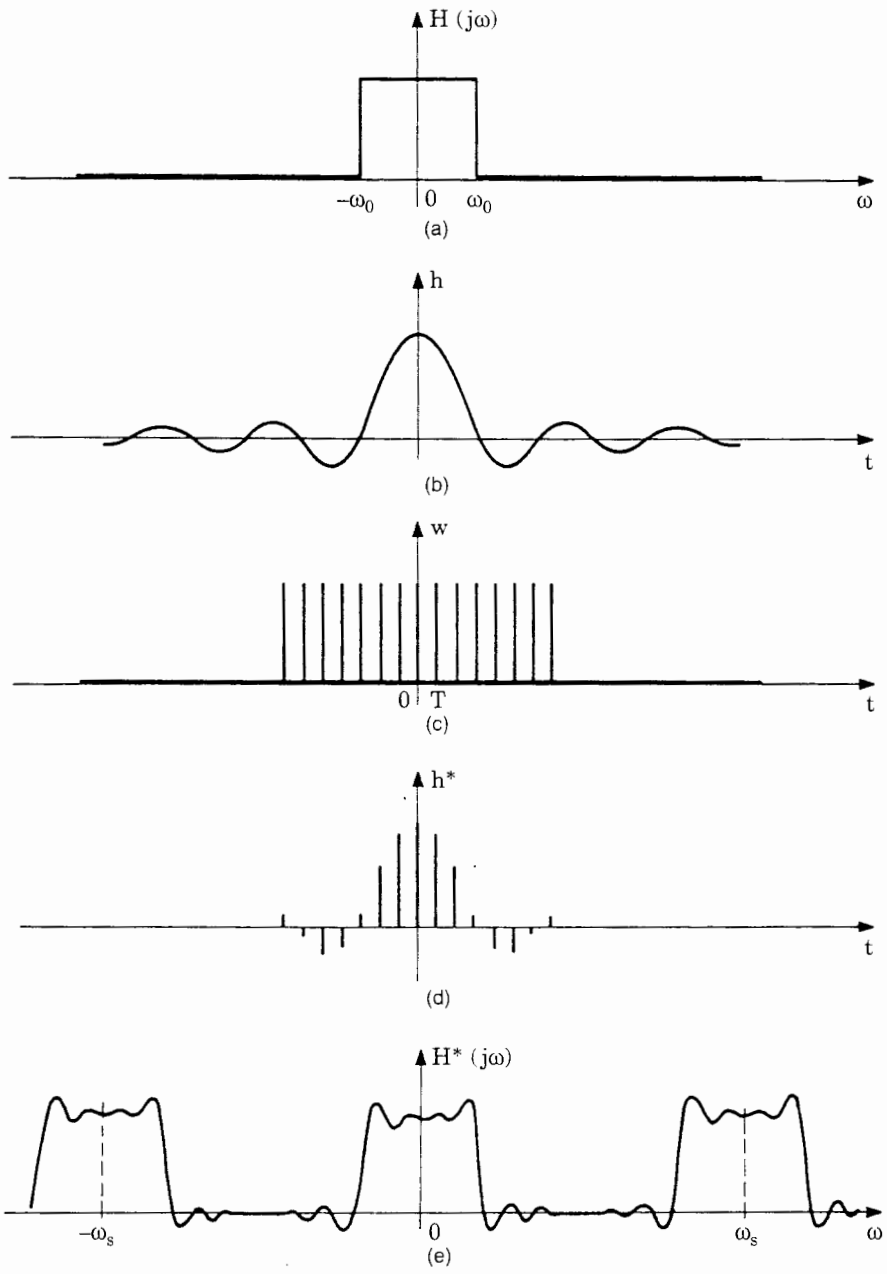


Figure C.10 This figure illustrates the design procedure for window FIR filters. (*a*) The design starts by defining the frequency response $H(j\omega)$ of an "ideal" analog filter. (*b*) The impulse response $h(t)$ of the "ideal" analog filter is calculated. (*c*) To get a finite and sampled impulse 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.

C.3.1 Window-FIR filters

When the window method is used, 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 that we set a goal that should be approximated within a given error specification. The "ideal" frequency response is plotted in Fig. C.10*a*. Next, the impulse response $h(t)$ of the ideal filter is calculated (Fig. C.10*b*). To get a finite impulse response, a portion of $h(t)$ is cut out using the window function $w(t)$ shown in Fig. C.10*c*. 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.10*d*. 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 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 [12,13,14]

$$W(j\omega) = \frac{\text{sinc}(\omega NT/2)}{\text{sinc}(\omega T/2)} \tag{C.27}$$

and 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

$$\Delta\omega_w = \frac{2\omega_s}{N} \tag{C.28}$$

and hence is inversely proportional to filter length N. The amplitudes of the side lobes decay slowly with increasing frequency, i.e., proportional to $1/f$. For an ideal low-pass filter, the result of the convolution $H(j\omega)^*W(j\omega)$ is shown in Fig. C.12. In the example shown, the sampling frequency has been chosen 4 times the corner frequency of the filter. The filter length has arbitrarily been chosen $N = 31$. Figure C.12*a* shows the frequency response $H^*(j\omega)$ with logarithmic amplitude scale, and Fig. C.12*b* with linear amplitude scale. [For calculation of $H^*(j\omega)$ it was assumed that the impulse response $h^*(t)$ is an even function of time, i.e., 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.12*a*]. Two effects of the convolution are easily recognized in Fig. C.12*a*: (1) the steep transition from pass-

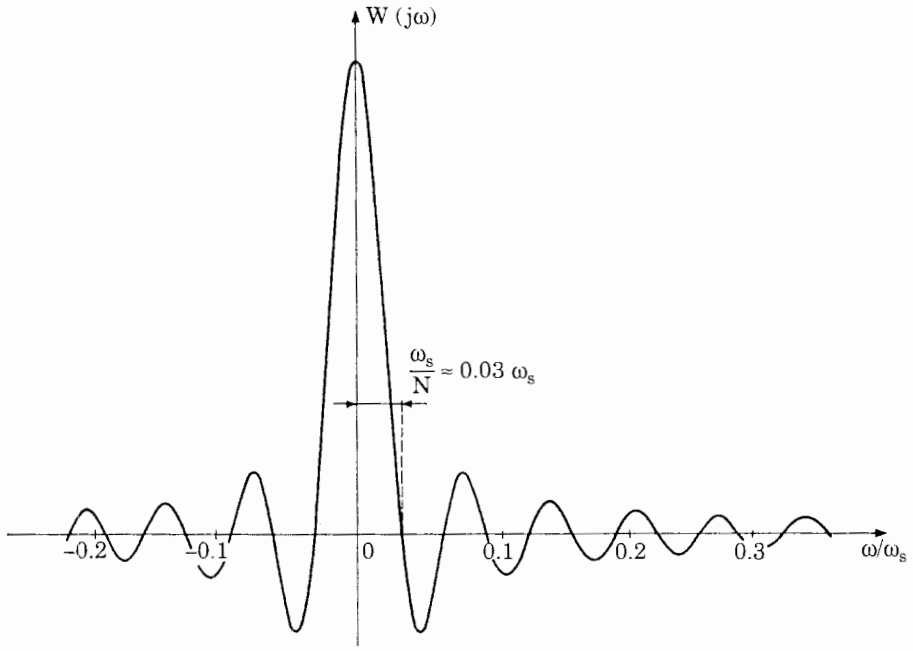
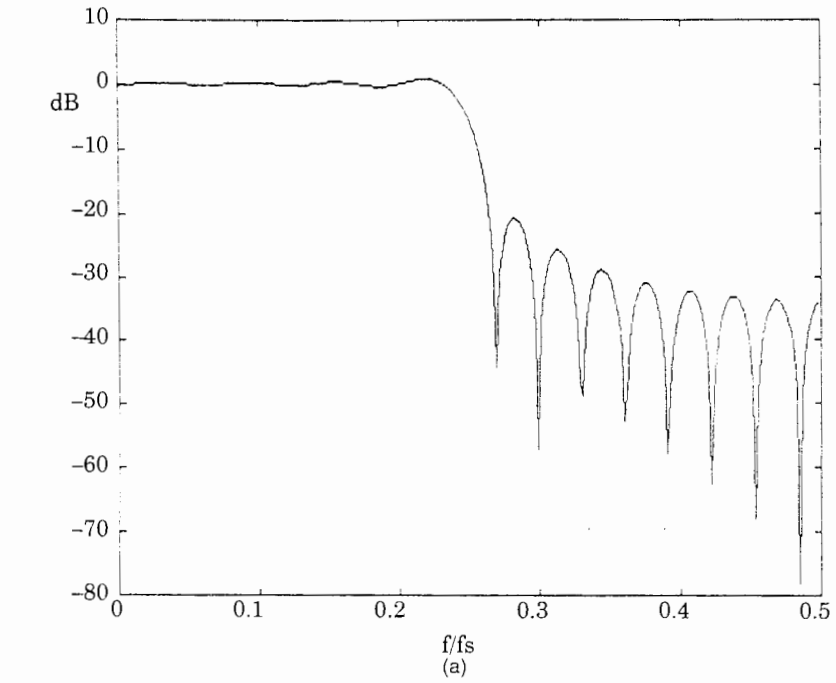


Figure C.11 Spectrum $W(j\omega)$ of the rectangular window $w(t)$ in Fig. C.10*c*. Only a portion of $W(j\omega)$ is shown here. The function is periodic in ω.

band 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.12*a*. Each ripple maximum on the linear amplitude scale corresponds to a damping minimum on the logarithmic scale. The first damping minimum (i.e., the first peak right from the transition in Fig. C.12*a*) 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 alternative window functions. Windows must be found, therefore, whose spectrum decays faster with increasing frequency. A great many different windows are known, e.g., the Hanning (also called von Hann), Hamming, Bartlett, Blackman, Kaiser, and flat-top windows.[12,13,14,19] Each of these windows has its particular pros and cons.



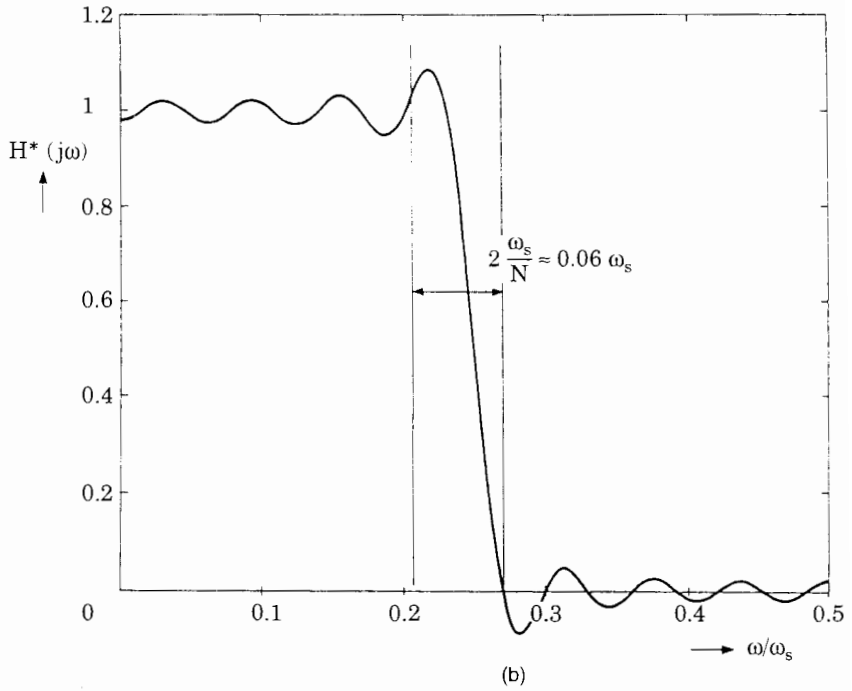


Figure C.12 The frequency response $H^*(j\omega)$ of the FIR filter is obtained by the complex convolution $H(j\omega)^*W(j\omega)$. (*a*) Frequency response $H^*(j\omega)$ on a logarithmic scale. (*b*) Linear plot of $H^*(j\omega)$.

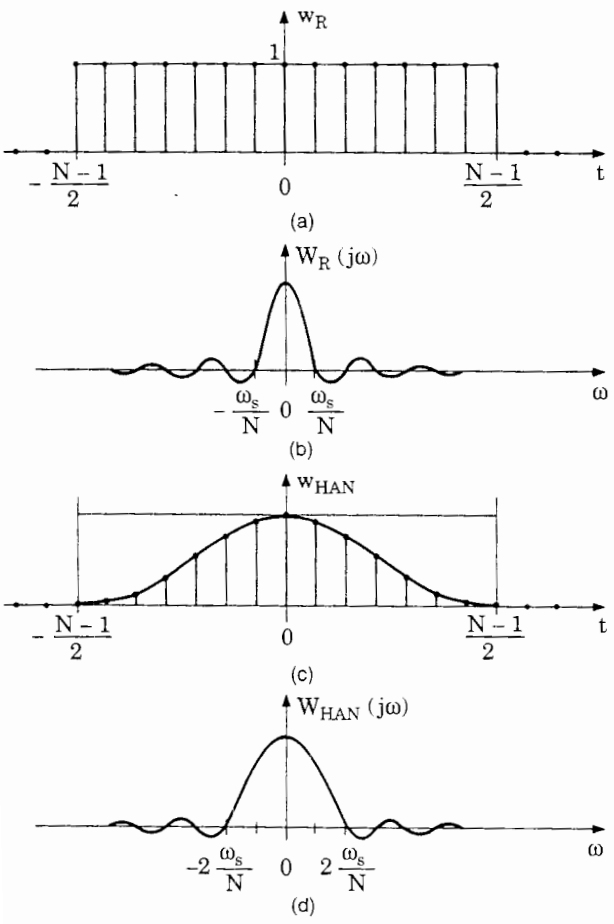


Figure C.13 Comparison of different window functions and their spectra. (*a*) The rectangular window $w_R(t)$. (*b*) The spectrum $W_R(j\omega)$ of the rectangular window. (*c*) The Hanning window $w_{HAN}(t)$. (*d*) The spectrum $W_{HAN}(j\omega)$ of the Hanning window.

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.13*a* shows once more the function of the rectangular window, which is now denoted w_R. Figure C.13*b* shows again its spectrum $W_R(j\omega)$. In Fig. C.13*c*, the Hanning window is plotted.

Its envelope is simply a cosine function; for this reason the Hanning window is often referred to as *raised cosine window*. Figure C.13*d* 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, as it widens the transition region (refer to Fig. C.12) by a factor of 2. The advantage of the Hanning window is in the fact, however, that the side lobes (refer to Fig. C.13*d*) decay much faster with increasing frequency, i.e., 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

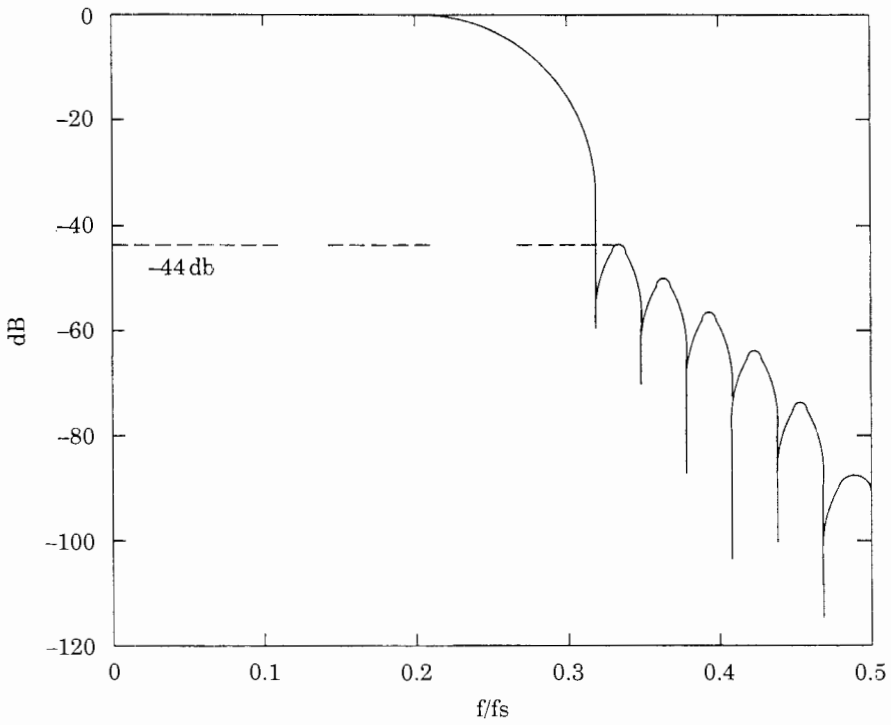


Figure C.14 Frequency response of a low-pass FIR filter, which uses a Hanning window.

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 largest damping minima with shortest filter lengths.[12] 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.

The FIR filters designed with the Parks-McClellan algorithm show up constant ripple in both stopband and passband, hence 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.

C.3.2 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 of 0 to 1 kHz, rejecting frequencies in the band from 1 to 4 kHz, passing frequencies

from 4 to 6 kHz, etc. When used to design a differentiator, the frequency response of a band-limited ideal differentiator[12] 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,[13] which is a filter having a gain of 1 and a constant phase shift of 90° 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 (Fig. C.15*a*). Next, transition width and ripple amplitude are specified by the boundary plot of Fig. C.15*b*. The FIR filter is required to have constant ripple ε_1 in

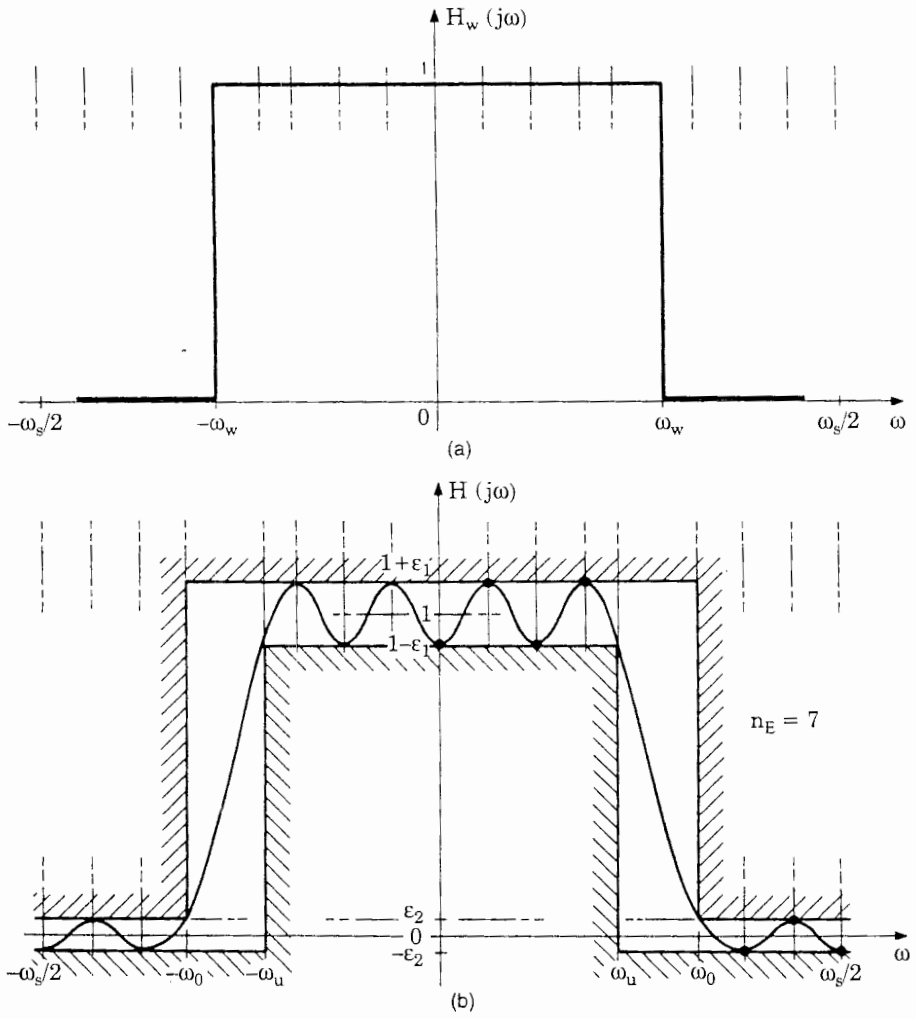


Figure C.15 This figure illustrates the design procedure for the Parks-McClellan algorithm. (*a*) The frequency response $H(j\omega)$ of an ideal analog filter is specified. (*b*) A boundary plot specifies the deviations of the actual filter from the ideal. In this plot, transition width and maximum ripple in the passband and stopband are defined.

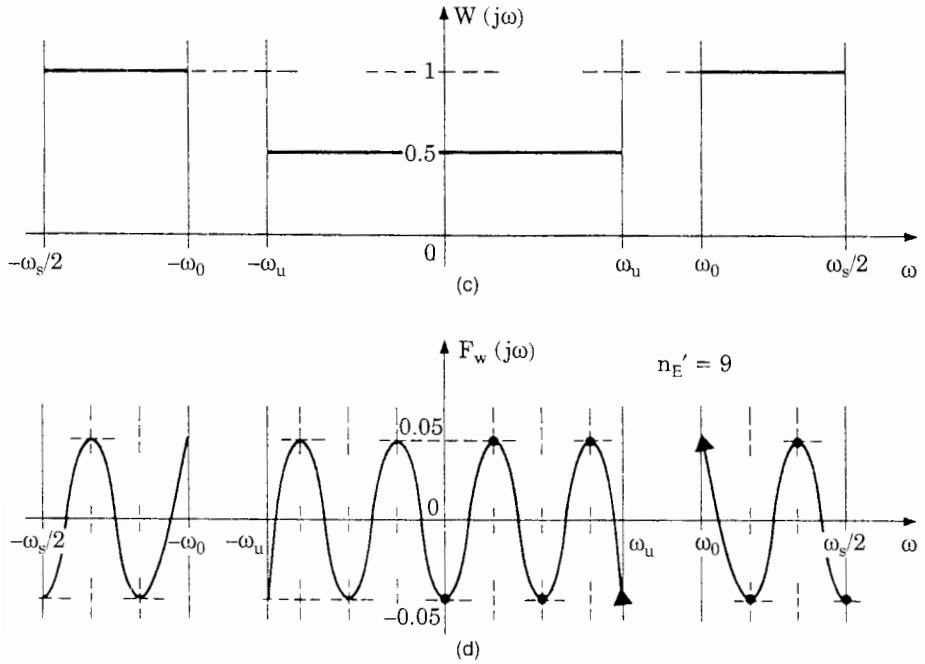


Figure C.15 (*Continued*) (*c*) Because different amounts of ripple are tolerated in passband and stopband, a weighting function $W(j\omega)$ is introduced. (*d*) The weighted error curve $F_w(j\omega)$ is given by the product of absolute error (from Fig. C.15*b*) and weighting function $W(j\omega)$ (from Fig. C.15*c*).

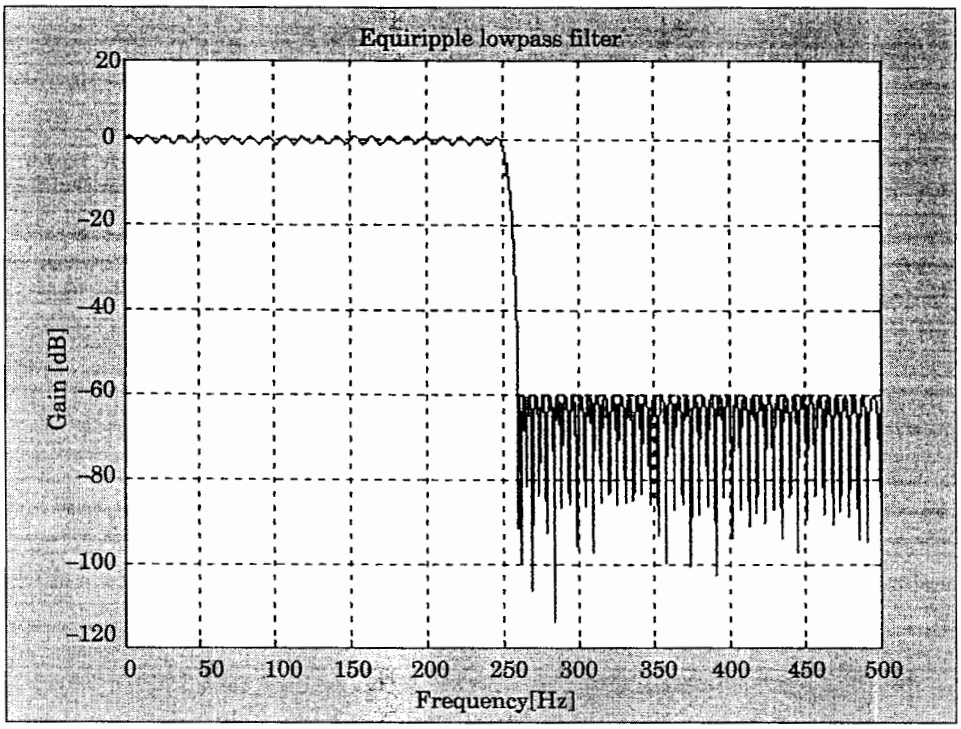
the passband and constant ripple ε_2 in the stopband. The actual frequency must lie entirely within the boundaries, as indicated in the drawing. The (unweighted) error function of the FIR filter is defined to be the difference between ideal and actual frequency response. Consequently, the (unweighted) error function has maxima and minima of $\pm\varepsilon_1$ in the passband and of $\pm\varepsilon_2$ in the stopband. Normally, it is desired that ε_2 be much smaller than ε_1. If we wish the FIR filter to have minimum damping of 60 dB in the stopband, for example, ε_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 in 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.15*c*. In our example, the allowed ripple in the stopband is half the ripple in the passband; hence the weighting function is chosen 1 in the stopband and 0.5 in the passband. Finally, Fig. C.15*d* 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.15*d*.

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

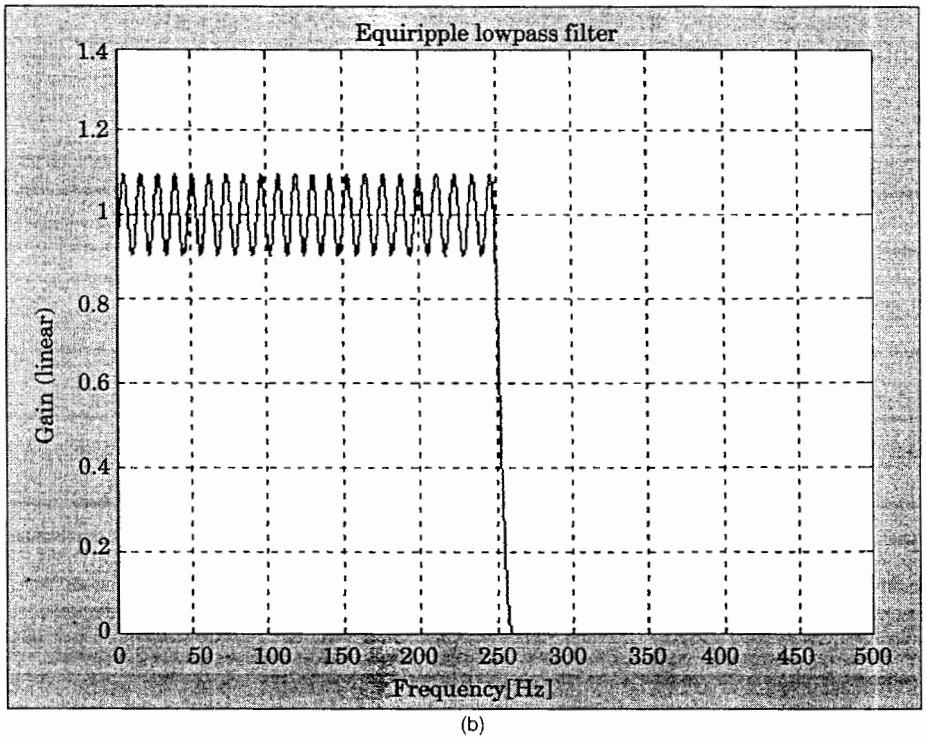
number of extrema (i.e., minima plus maxima) of the weighted error function in the frequency range 0 to f_n (Nyquist frequency) is around $N/2$. (The exact figure depends somewhat on the type of FIR filter to be realized; we will not go into details here.[13]) The Parks-McClellan algorithm starts with an initial guess of the locations of the extrema of $F_w(j\omega)$; see the filled dots in Fig. C.15*d*. 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.[13] (When N points of a curve are specified, it is relatively simple to determine a polynomial of degree $N - 1$ that exactly goes through these points; this kind of polynomial is called a Lagrange polynomial and is often used for the interpolation of functions.) If the initial guess had be 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 to determine 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 so many passes, until the extrema found in the Kth 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 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 $h(0), h(1), \ldots, h(N - 1)$, where N is the filter length. Figure C.16 is an example of an FIR low-pass filter of length $N = 180$ that has been designed by the signal processing toolbox of Matlab.[35] This filter has the following specifications:

- Sampling frequency $f_s = 1000\,\text{Hz}$
- 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 very low stopband ripple and a rather narrow transition bandwidth (10 Hz). Figure C.16*a* 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 sclae (Fig. C.16*b*). Also, here we note that the passband ripple exactly meets the design goal (±0.1).



(a)



(b)

Figure C.16 Equiripple low-pass filter designed with Matlab.[35] Maximum ripple in the passband is 0.1 (0.8 dB), maximum stopband ripple is 0.001 (–60 dB), and transition width is 10 Hz. To meet these specifications, a filter length $N = 180$ is required: (*a*) Gain versus frequency on a semilogarithmic scale. (*b*) Gain versus frequency on a linear scale.

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 filter, e.g., IIR filters and FIR filters, with the window method and with the Parks-McClellan algorithm.[25,35] When using such a software tool, the designer will enter first 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.

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 Behavior 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.
10. Rohde, Ulrich L.: "Low-Noise Frequency Synthesizers Using Fractional *N* Phase-Locked Loops," *r. f. design,* January/February 1981.
11. Rohde, Ulrich L.: *Digital PLL Frequency Synthesizers, Theory and Design*, Prentice Hall, Englewood Cliffs, NJ, 1983.
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*, Prentice Hall, Englewood Cliffs, NJ, 1988.
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 électrique*, 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, App. D.
24. Selle, D.: "Theoretische Grundlagen, Dimensionierung und charakteristische Kenngrössen von PLL-Schaltungen (Phasenregelkreisen)," 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 (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*, Copyright 1977, 1978 by Digital Equipment Corporation, Bedford, MA 01730.
29. Larimore, W. E.: "Synthesis of Digital Phase-Locked Loops," in *EASCON Rec.*, October 1968, pp. 14–20.
30. Langston, J. Leland: "μC 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 (see 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, April 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*.
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).
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, McGraw-Hill, New York, 3d ed., 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 for 74HCT9046A, can be downloaded from *http://philips-logic.com/products/plls/9046*.
52. "CMOS phase-locked loops: 74HC(T)4046A/7046A and 74HCT9046A, HCMOS Designer's Guide—Advance Information," revised edition: June 1995, including diskette (3.5 inch) with design program (DOS application), document order number 9397 750 00078, Philips Semiconductor, 811 East Arques Avenue, Sunnyvale, CA 94088-3409.

Index

*Boldface numbers indicate where topics are discussed in full.

ABOUT THE AUTHOR

ROLAND E. BEST is the founder of Best Engineering and a world-renowned authority on phase-locked loops, circuit design, and microprocessor applications. Mr. Best resides in Basel, Switzerland.

ABOUT THE AUTHOR

ROLAND E. BEST is the founder of Best Engineering and a world-renowned authority on phase-locked loops, circuit design, and microprocessor applications. Mr. Best resides in Basel, Switzerland.