Copyrighted Material

John Rogers
Calvin Plett
Foster Dai

INTEGRATED CIRCUIT DESIGN FOR HIGH-SPEED FREQUENCY SYNTHESIS

Copyrighted Material

Integrated Circuit Design for High-Speed Frequency Synthesis

For a complete listing of recent titles in the *Artech House Microwave Library,* turn to the back of this book.

Integrated Circuit Design for High-Speed Frequency Synthesis

John Rogers
Calvin Plett
Foster Dai

ARTECH HOUSE
BOSTON | LONDON
artechhouse.com

Library of Congress Cataloging-in-Publication Data
Rogers, John (John W. M.)
Integrated circuit design for high-speed frequency synthesis / John Rogers, Calvin Plett, Foster Dai.
p. cm.—(Artech House microwave library)
Includes bibliographical references and index.
ISBN 1-58053-982-3 (alk. paper)
1. Very high speed integrated circuits—Design and construction. I. Plett, Calvin. II. Dai, Foster. III. Title. IV. Series.

TK7874.7.R65 2006
621.3815—dc22 2005044873

British Library Cataloguing in Publication Data
Rogers, John (John W. M.)
Integrated circuit design for high-speed frequency synthesis.—(Artech House microwave library)
1. Integrated circuits—Design and construction 2. Frequency synthesizers
I. Title II. Plett, Calvin III. Dai, Foster
621.3’815

ISBN-10: 1-58053-982-3

Cover design by Yekaterina Ratner

© 2006 ARTECH HOUSE, INC.
685 Canton Street
Norwood, MA 02062

All rights reserved. Printed and bound in the United States of America. No part of this book may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage and retrieval system, without permission in writing from the publisher.

All terms mentioned in this book that are known to be trademarks or service marks have been appropriately capitalized. Artech House cannot attest to the accuracy of this information. Use of a term in this book should not be regarded as affecting the validity of any trademark or service mark.

International Standard Book Number: 1-58053-982-3
Library of Congress Catalog Card Number: 2005044873

10 9 8 7 6 5 4 3 2 1

Contents

Preface

This book started as notes for a graduate course on integrated synthesizers. After starting to develop this course, we found that while there were excellent books on synthesizer design and excellent books on integrated circuit (IC) design, there were none dealing exclusively with fully integrated synthesizer design. As a result, we decided that there was a need for a book that dealt with frequency synthesizers and the circuits that are used to implement them in modern IC processes. Thus, the course notes were further developed into this book, which is based on the combined experience of the authors' teaching, research, and consulting activities in this area.

In spite of our best intentions, there will certainly be errors in this book (we are human). We will be keeping a list of errors on a Web page, and we encourage you to make use of it and to inform us of any additional errors not yet listed on this page, making this reference of greater value. The Web page is at http://www.doe.carleton.ca/~cp/synth_errata.html.

We need to thank a lot of people who have helped us in the creation of this book. We thank the students who put a huge effort into proofreading the book, struggling through the examples and doing assignments based on the information in this book. Specifically, we would like to thank Mark Houlgate, James Chiu, Harpreet Panesar, Charles Berndt, Ziad El-Khatib, Siva Kumar, Daniel Olszewski, Steve Penney, Ghyslain Gagnon, Tony Forzley, Peter Chyurlia, Jorge Aguirre, Faisal Saleh, Samer Abielmona, Fiona Shearer, Celine Fletcher, Steve Knox, Justin Abbott, Paul Laferriere, Peter Popplewell, Victor Karam, Vincent Karam, Hao Shang, and Travis Lovitt. Many other people have also edited part or all of this work, including Ardeshir Namdar, Matt Wilbur, Ken Townsend, and Mike Toner. Garry Tarr and John Knight also provided technical assistance on a number of chapters that went beyond simply pointing out errors. Sandi Plett has contributed a great deal in helping to format this manuscript and to get it ready for publication.

John and Foster would like to acknowledge their former colleagues at Cognio Inc.: Dave Rahn, Ted Hokey, Mark Cavin, Neric Fong, Richard Griffith, Sifen Luo, Bob Macuccie, David Moore, Jose Macedo, Bill Seed, Gary Sugar, Mike Toner, Fan Qing, and Jam Zhou. Under Dave's leadership, this group successfully designed a fully integrated multiple input, multiple output (MIMO) wireless local area network (WLAN) transceiver RFIC while this book was under development. As a result, much of the fundamental understanding of synthesizers gained from this project ended up in this manuscript.

Acknowledgments for John Rogers

This book would not have been possible if it were not for the support of a number of people in my life. First, I would like to thank Calvin and Foster, my coauthors and two of my best friends, for allowing me to infect them with my enthusiasm for this project. It goes without saying that, without them, it would have never have come together. I owe a great deal of my professional success to the confidence that Dave Rahn showed in me over the four years I worked for him. If it weren't for Dave, I would never have met Foster, and this project would not have happened. Dave, if only there were more people like you in the world, it would be a much nicer place to live in. I am very proud to count you among my best friends.

While this work was written, I was fortunate to have the support of a number of people. Thanks go especially to my parents, Ann Gibson and John C. Rogers, as well as my close friends and colleagues Neric Fong, Huyen Le, Pascal deWitt, and Mike Bonito for support in a lot of ways. Professionally, I had the opportunity to learn something about synthesizer design from Mark Cavin.

Finally, I would like to dedicate the part of this work that I penned to Jing Chen. Writing these pages kept me sane while I was waiting for her to come into my life.

Acknowledgments for Calvin Plett

I would like to thank my two coauthors, John and Foster. Their seemingly boundless energy and superhuman drive made this book happen. I am also grateful for support and encouragement from students, staff, and faculty at Carleton University, some of whom have been mentioned specifically above. I would like to thank all of my industrial colleagues, especially Mark Cloutier for the interesting, thought-provoking discussions.

I would like to acknowledge and thank my colleagues and former students at Carleton University who did some of the pioneering work in the development of frequency synthesizers. In particular, some of the first work on the use of sigma-delta modulators applied to frequency synthesizers and on the modulation of synthesizers was done by Tom Riley, Tad Kwasniewski, Miles Copeland, Walt Bax, and Norm Filiol.

Finally, to Sandi for always being there, thanks.

Acknowledgments for Foster Dai

I would like to thank the two coauthors of this book. John and Calvin, without your expertise, enthusiasm, and hard work, this book would never have come to life on time.

Much of my knowledge in frequency synthesis comes from my work at Hughes Network Systems, where I fortunately met a few real experts in this area. I would particularly like to thank Lawrence Blue. Larry, without your encouragement and support, I would never have had the opportunity to work in the area of IC design.

I would especially like to thank Chris Clark, Steve Rosenbaum, Steven Johns, Olu George, Thomas Jackson, George Eapen, and many others for their enlightening discussions and valuable help. I am very grateful to my colleagues at Auburn University, Richard Jaeger, Dave Irwin, Guofu Niu, John Wu, Wayne Johnson, Charles Stroud, and many others for valuable discussions and support in various ways. A special debt is owned to my graduate students at Auburn University, Xuefeng Yu, Vasanth Kakani, Dayu Yang, Lakshmi Chimakurthy, Malinky Ghosh, and Raja Sandireddy, for their contributions to this book through their research with me. I am very appreciative of the collaborations with many respected experts in VLSI engineering. In particular, I would like to thank John Cressler at the Georgia Institute of Technology, Lawrence Larson at the University of California, San Diego, Ben Blalock at the University of Tennessee, Mohammad Mojarradi at the NASA Jet Propulsion Laboratory, and Yin Shi at the Chinese Academy of Science.

I would like to thank my parents, Yonggang Dai and Jiazhen Li, my wife, Wendy, my son, Darren, and my daughter, Daisy, for their love and patient support throughout my education and career development. Finally, I would like to acknowledge Dr. Ruying Yin, my mentor since middle school, for his guidance and encouragement.

CHAPTER 1

Introduction

1.1 Introduction to Frequency Synthesis

Over the past few decades, there has been an incredible growth in the electronics industry. Electronic equipment from cell phones to personal computers now affects a great many aspects of our everyday lives. In order to make these devices ubiquitous, they must be low cost and compact. This increasingly makes integrated circuit (IC) technology the obvious choice for many of these products. As these products become more complex and more integrated, there is a greater need than ever to develop methods for designing frequency synthesizers in IC technologies, and that will be the focus of this book.

Frequency synthesis is, in general, concerned with the generation of periodic waveforms. This seems like a simple task, but it in fact takes up significant area and effort in many ICs. Generating good-quality periodic waveforms that meet many system requirements is not a trivial task. So, even though in system block diagrams the frequency synthesizer is typically only shown by one tiny box in the corner, the effort expended in designing this block should not be overlooked!

Many of the techniques used in frequency synthesis in this book actually predate the IC. Thus, in the past, there have been two classes of books: those that deal with IC-based circuit design and those that deal with frequency synthesis. In this book, the aim will be to combine these two topics and to discuss mainly the elements of frequency synthesis that are important for IC design. Likewise, the aim will be to present circuit implementations for the building blocks that are best suited to IC technology. The technology of choice in this book for the implementation of frequency synthesizers will be primarily complementary metal oxide silicon (CMOS) and, to some degree, silicon germanium (SiGe) bipolar CMOS (BiCMOS). It should be noted that while the world is progressively moving more to CMOS-only implementations, there is still a very active group of designers working in SiGe, which in many respects is a superior technology.

The rest of this chapter is intended to give the reader an overview of frequency synthesizer applications, to show some common systems where frequency synthesizers are needed, and to highlight some of the issues that the designer must consider when designing a frequency synthesizer to work in these systems.

1.2 Frequency Synthesis for Telecommunications Systems

A major group of applications that require frequency synthesizer circuits are telecommunications systems. Frequency synthesis provides the means by which a communication or broadcast channel can be selected. Examples of such applications are cell phones, wireless local area networks (WLANs), pagers, and many more. All these devices require radios. A radio is a system that enables the flow of information from one place to another, usually through the air or a cable. A basic block diagram of a typical superheterodyne radio transceiver using air as a medium is shown in Figure 1.1. Modulated signals are transmitted and received at some frequency by an antenna. If the radio is receiving information, then the signals are passed from the antenna to the receiver (Rx) part of the radio. The signals are then passed through a filter to remove interference in frequency bands other than the one of interest. The signal is then amplified by a low noise amplifier to increase the power in weak signals, while adding as little noise (unwanted random signals) as possible. The spectrum is further filtered by an image filter and then downconverted by a mixer to an intermediate frequency (IF).

The mixer (also sometimes called a multiplier) mixes the incoming spectrum of radio-frequency (RF) signals with the output from the first frequency synthesizer. The desired output of the mixer is the difference between the RF and local oscillator (LO) frequencies. The LO can be either low-side injected (the LO is at a frequency less than the RF frequency) or high-side injected (the LO is at a frequency greater than the RF frequency). For low-side injection, the IF frequency is given by

$$f_{\mathrm{IF}} = f_{\mathrm{RF}} - f_{\mathrm{LO}} \tag{1.1}$$

For high-side injection, it is given by

$$f_{\mathrm{IF}} = f_{\mathrm{LO}} - f_{\mathrm{RF}} \tag{1.2}$$

In a traditional radio, the IF stage is at a fixed frequency; therefore, the synthesizer must be made programmable so that it can be tuned to whatever input frequency is desired at a given time. Note that an input signal at an equal distance

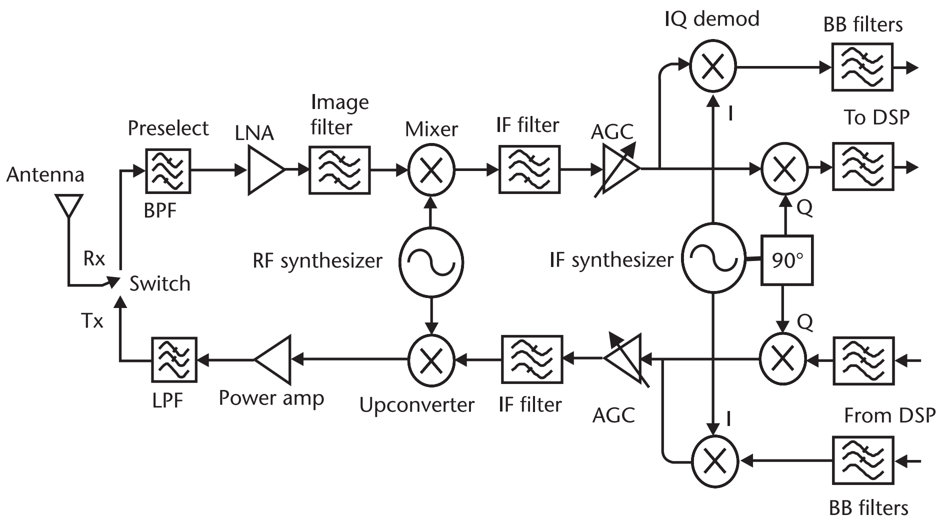


Figure 1.1 A typical superheterodyne radio transceiver.

from the LO on the other side from the desired RF signal is called the *image signal* because a signal at this frequency, after mixing, will be at the same IF as the desired signal. Therefore, the image signal cannot be removed by filtering after mixing has taken place. Thus, an important job of the RF filters is to remove any image signals before such mixing action takes place. This is illustrated in Figure 1.2.

After mixing to an IF, additional filtering is usually performed. At the IF, unwanted channels can be filtered out, leaving only the channel of interest, since its frequency is now centered at the IF. Usually, automatic gain control (AGC) amplifiers are also included at the IF. They adjust the gain of the radio so that its output amplitude is always constant. Once through the AGC, the signals are downconverted a second time to baseband (the signals are now centered around dc or zero frequency). This requires a second frequency synthesizer that produces both 0° and 90° output signals at the IF frequency. Two paths downconvert the signals into in-phase (I) and quadrature-phase (Q) paths (I and Q paths are separated by a phase shift of 90°). By using this technique, the incoming phase of the RF signal does not need to be synchronized to the phase of the LO tone. The I and Q signals are then passed through baseband filters and into the back end of the radio, which removes the rest of the unwanted channels. The back end, or digital signal processing (DSP) circuitry, then converts the signals back into digital information.

The transmitter works in much the same way, except in reverse. The DSP circuitry produces signals in quadrature. These signals are then filtered and upconverted to an IF frequency and again passed through an AGC and filtered. They are then upconverted to the RF frequency by the mixer. Note that in this case, the multiplying or mixing action of the mixer is used to generate sum, rather than difference, products if the LO is low-side injected. Thus, for low-side injection, the RF frequency is given by

$$f_{\mathrm{RF}} = f_{\mathrm{LO}} + f_{\mathrm{IF}} \tag{1.3}$$

If the LO is high-side injected, the frequency of the RF signal is given by

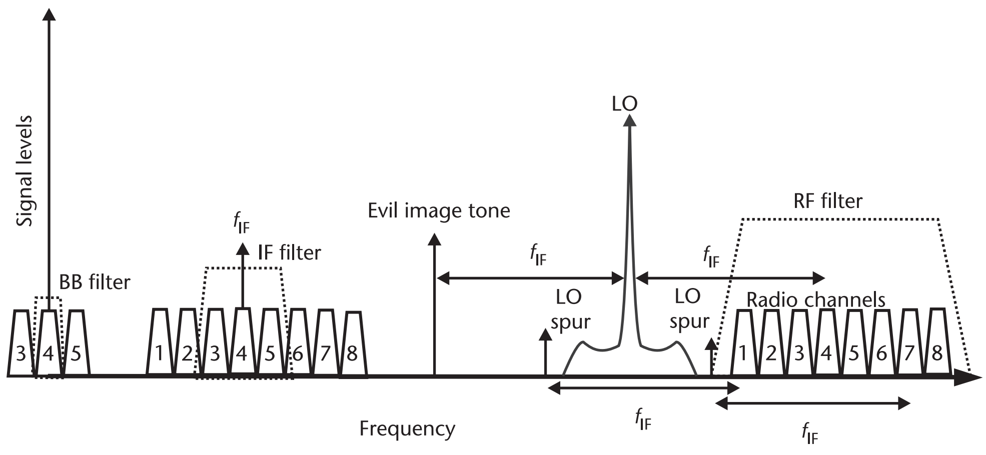


Figure 1.2 Radio receiver frequency plan and some LO issues.

$$f_{\mathrm{RF}} = f_{\mathrm{LO}} - f_{\mathrm{IF}} \tag{1.4}$$

Once upconverted to RF, the signal is then passed through a power amplifier to increase the power of the signals and is then radiated by the antenna into the air. The power level must be high enough so that the signal can be detected at the maximum distance dictated by the system specifications.

The synthesizers are a very important part of the radio. They must meet very demanding specifications for their performance. These could include

- *Tuning range:* The RF synthesizer must be able to cover all frequencies necessary to downconvert every channel of interest.
- *Purity of the output tone:* Generally, in addition to the desired tone, there will be additional unwanted frequencies around the desired frequency. The spreading of the tone power in the frequency domain around the desired frequency is often called *phase noise*. In the time domain, this can be thought of as *phase jitter* (variations in the phase of the waveform at any given instant).
- *Freedom from spurs:* The waveform must be free of spurs, or spurious frequency components, which are frequency components other than the one desired. Specifications for spurs in a frequency synthesizer usually require that the amplitude of the spurs be at least a specified number of decibels lower than the amplitude of the desired carrier. The effect of a spur is illustrated in Figure 1.2, where the LO mixes channel 4 (the desired channel) down to the IF. LO spurs separated from the LO by Δf could mix channel 7 and channel 1 (both unwanted channels and both at a frequency offset of Δf from the desired channel) on top of channel 4 at IF, corrupting it.
- *Amplitude:* The amplitude of the waveform that appears at the mixer must be sufficient to drive the mixer successfully. This can be difficult at multigigahertz speeds, especially since the synthesizer and mixer are often separated by many millimeters of on-chip transmission line.
- *Step size:* Adjacent programmable frequencies of the synthesizer must be separated by no more than the channel spacing of the radio.
- *Settling time:* The synthesizer must be able to change its frequency from one channel to another in a given time, or else data may be lost.
- *Acquisition time:* After turning on the frequency synthesizer, its frequency must move to a programmed frequency in a given amount of time.
- *I and Q matching:* The phase difference between the I and Q channels ideally will be 90°. If it deviates from this, it may not be possible to decode the data in the I and Q channels properly.
- *Power consumption:* Generally, all the circuits in a synthesizer must consume less than a specified amount of current and power.
- *Synthesizer pulling (chirp):* If other circuitry is turned on or off, sometimes this will cause an instantaneous change in the frequency of the synthesizer. This undesired property is sometimes called *chirp* and must be below a certain maximum frequency change and must settle back to the correct frequency within a specified amount of time.

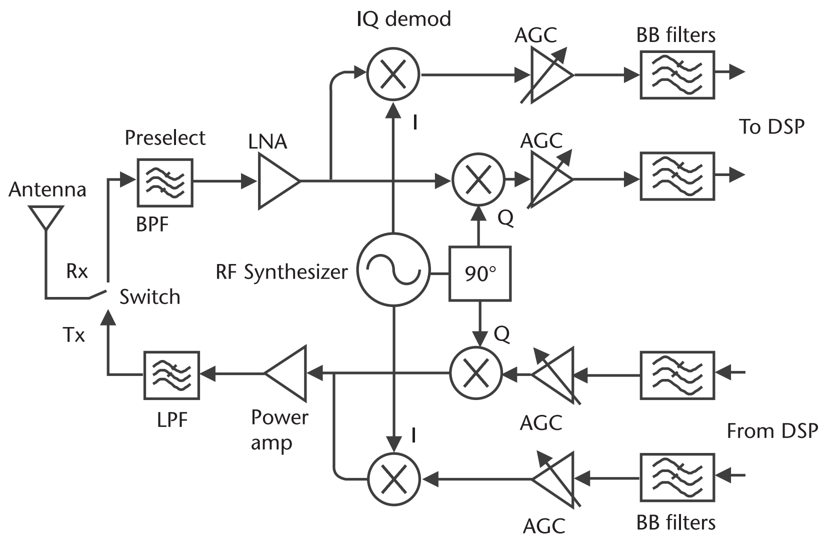
The requirements for synthesizers clearly depend on the communication system. For example, the Global System for Mobile Communications (GSM) has very stringent noise specifications, so synthesizers must have very low phase noise. On the other hand, the Bluetooth standard has more relaxed noise specifications on account of its being a short-range, low-power, low-cost system. Time division multiple access systems, which may change frequencies upon changing between transmit and receive, have requirements for the synthesizer switching speed and settling time.

The superheterodyne radio just presented is not the only possible architecture. A common variation is the direct downconversion architecture shown in Figure 1.3(a). In this architecture, the IF stage is omitted and the signals are converted directly to dc. For this reason the architecture is sometimes called a *zero-IF* or *direct-conversion* radio. The direct-conversion transceiver saves the area and power associated with a second synthesizer, as well as some other components. However, it is inferior in a number of ways, not the least of which is that generating I and Q signals from a synthesizer at higher frequencies is much more difficult than doing so at the IF. Another architecture that is a compromise between the superheterodyne and the direct-conversion transceiver is called a *walking IF* architecture, shown in Figure 1.3(b). This architecture derives the IF LO by dividing the RF LO by some fixed number. As a result, the IF frequency is not fixed but "walks" in step with a fraction of the frequency of the RF LO. This transceiver with walking IF still has many of the advantages of the superheterodyne radio (although it is not possible to filter as well at IF), but it also removes the need for the extra synthesizer, potentially reducing layout area and power dissipation.

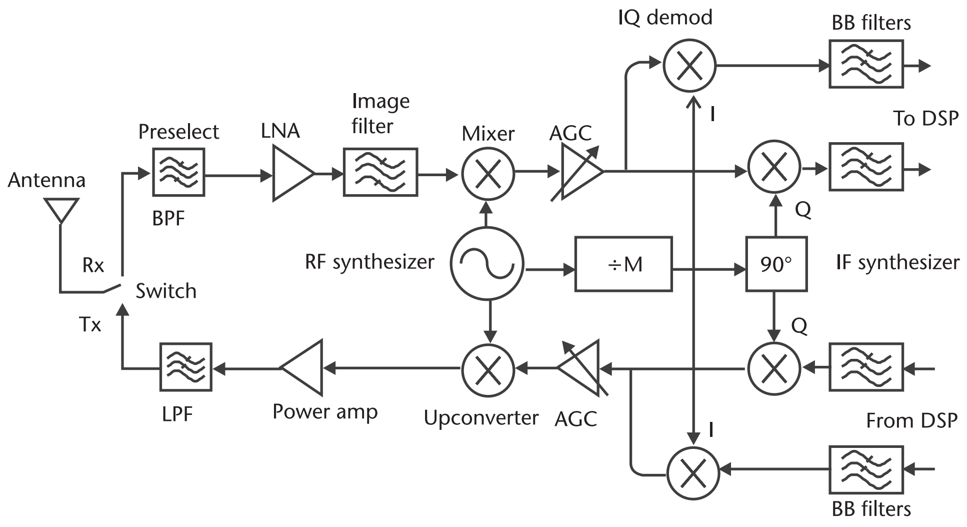
Obviously, in only a few pages, it would be impossible to give a complete account of radio architectures. The interested reader is referred to [1–14] for more information on this topic.

1.3 Frequency Synthesis for Digital Circuit Applications

The realization of digital circuits is probably the most common use of IC technology in the world. A well-known example of a digital IC circuit is the central processing unit of a computer. Digital circuits work exclusively on performing mathematical, logical, and control operations. A clock that is often generated by a frequency synthesizer triggers each mathematical operation and corresponding change in state of the circuit. A generic block diagram for a digital circuit is shown in Figure 1.4. The state of the circuit is stored in a device called a flip-flop, shown as *FF* in Figure 1.4. The output of all the flip-flops in the system (whether they currently put out a logical zero or a one) determines the state of the system. In this discussion, it will be assumed that the flip-flop is clocked by the rising edge, although implementing it to be triggered on the falling edge is also possible. On each rising clock edge, the flip-flop passes the signal present at its input to the output and, thereby, changes state. The current outputs of flip-flops, as well as external inputs, passed through logical functions, determine what the next state of the system will be.



(a)



(b)

Figure 1.3 Common variations on radio architectures: (a) a direct downconversion radio, and (b) a walking IF radio.

The clock determines the speed at which the circuit operates. If this clock is set too fast then the circuit will not work properly. Considerations in determining the speed that a digital circuit can be clocked will be described next.

After a flip-flop experiences a rising edge of the clock, its output (Q) will change state, but this will happen in a finite amount of time $T_{\text{clk-to-q}}$.

It will take a finite amount of time after the flip-flop output has changed state for that change to propagate through the system's logic to arrive at the inputs of the flip-flops in the following stage. This time is the *propagation delay* T_{pd}.

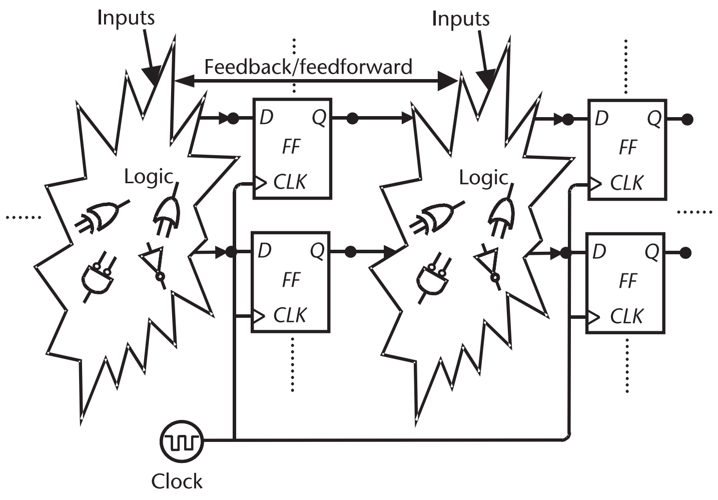


Figure 1.4 A generic digital circuit.

The data input (D) to the flip-flop must be stable for a time of at least T_{setup} prior to the clock (C) transition, as shown in Figure 1.5. This is known as the *setup time,* or T_{setup}, constraint. Setup time violation should be checked under the maximum T_{pd}. The data input (D) to the flip-flop must be held for a period at least equal to T_{hold} after the clock (C) transition, as shown in Figure 1.5. This is known as the *hold time,* or T_{hold}, constraint. Hold time violation should be checked under the minimum T_{pd}. Thus, if a digital circuit is driven by an ideal clock, then the speed of the clock is limited by

$$T \geq T_{\text{clk-to-q}} + T_{\mathrm{pd}} + T_{\mathrm{setup}} \tag{1.5}$$

One of the main challenges with modern digital circuits in regard to generating the clock is synchronizing the phase of the clock as it is distributed across the chip.

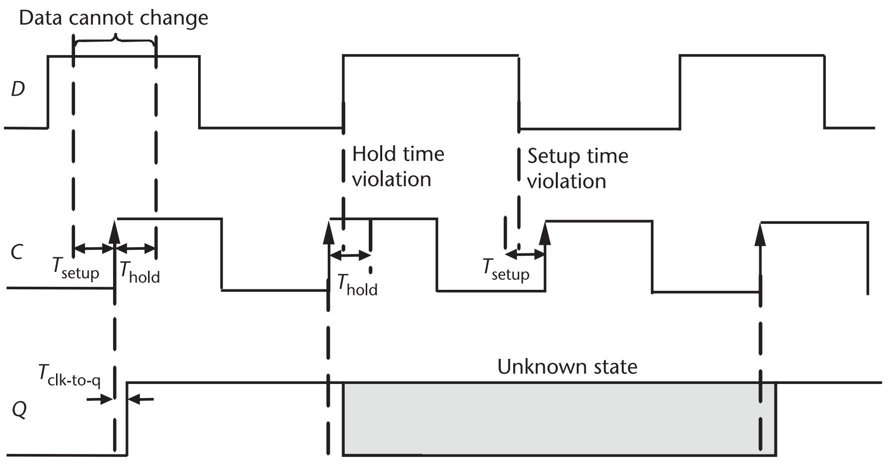


Figure 1.5 *D*-flip-flop setup and hold-time constraints.

Unlike telecommunications applications, where a synthesizer may drive only one or two mixers, in digital circuits, the clock may drive tens of thousands of flip-flops. All of these flip-flops need to change state simultaneously; thus, the clock must be distributed very carefully all over the chip. This may involve driving many tens of millimeters of interconnect at multigigahertz speed. Of course, some difference in delay across the chip is unavoidable. This is called *clock skew*. The time between the first flip-flop and the last flip-flop getting a clock edge is defined as T_{skew}. Clock skew must be added to (1.5) for setting the maximum speed of the circuit. Note that if the skew is too large, additional clock tree buffers are needed to retime the clock as it is distributed across the chip.

While spurs and jitter in phase are not directly as much of an issue in digital clocks as they are in other applications, it can still be an issue if the phase jitter is too high, as this will affect the maximum speed at which the circuit can operate. After a clock edge has triggered a change in state of the digital circuit, then all the circuits have until the next clock edge to settle out. If the clock edge arrives early, then all the operations must already have settled out by that point. This is illustrated in Figure 1.6. Thus, the peak jitter time T_{jitter} must also be added to the time for the minimum clock period of the digital circuit. Thus, with a nonideal clock, the worst-case minimum clock period T is

$$T \geq T_{\text{clk-to-q}} + T_{\text{pd}} + T_{\text{setup}} + T_{\text{skew}} + T_{\text{jitter}} \tag{1.6}$$

Note that digital design in general is a huge topic. For more information, see [15–19].

1.4 Frequency Synthesis for Clock and Data Recovery

There is another major class of communication circuits that transmit binary on-off key bit streams through optical fiber [20, 21]. The available information capacity

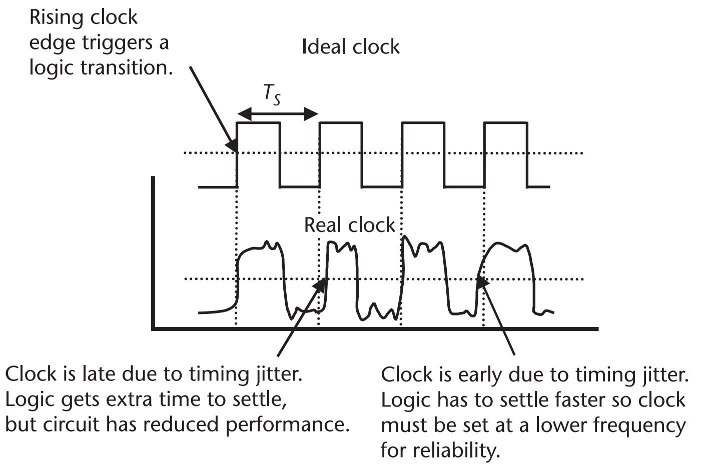

Rising clock edge triggers a logic transition.
Ideal clock
T_S
Real clock
Clock is late due to timing jitter. Logic gets extra time to settle, but circuit has reduced performance.
Clock is early due to timing jitter. Logic has to settle faster so clock must be set at a lower frequency for reliability.

Figure 1.6 Ideal versus real digital clocks with timing jitter and the resulting problems.

is proportional to the carrier frequency in any communication system. Due to the extremely high carrier frequency at infrared, wireline fiber optic communication has attracted tremendous attention over the past decade. Because of the Internet revolution, fiber networks, the major carrier of Internet backbone traffic, have boosted the information capacity to a level beyond our imagination not long ago. A typical broadband transceiver is shown in Figure 1.7. Here the binary on-off key modulated data (a stream of ones and zeros represented by light pulses) is transmitted through a fiber optic cable.

On the receiver side of the circuit, a polarization mode dispersion (PMD) compensation unit is used. This is used to compensate for PMD, which results in pulse broadening due to different light speed along different polarization directions in the optical fiber. Chromatic dispersion (CD) results in a spreading of frequency components due to nonuniform group delay in the fiber. To compensate for this, typically, a CD compensation unit is used. However, use of these blocks results in some attenuation. To counteract this attenuation, an erbium doped fiber amplifier (EDFA) can be used. The receiver EDFA also provides an overall gain control so that the receiver front end can operate in its proper dynamic range. The optically compensated signal is then detected by a photodetector (labeled PD in Figure 1.7) that produces a small amount of current proportional to the received optical signal. This current is then amplified and converted into a voltage by a transimpedance amplifier. Next, the signal is fed through an AGC amplifier or, alternatively, a limiting amplifier to further increase the signal strength. Since there is no amplitude

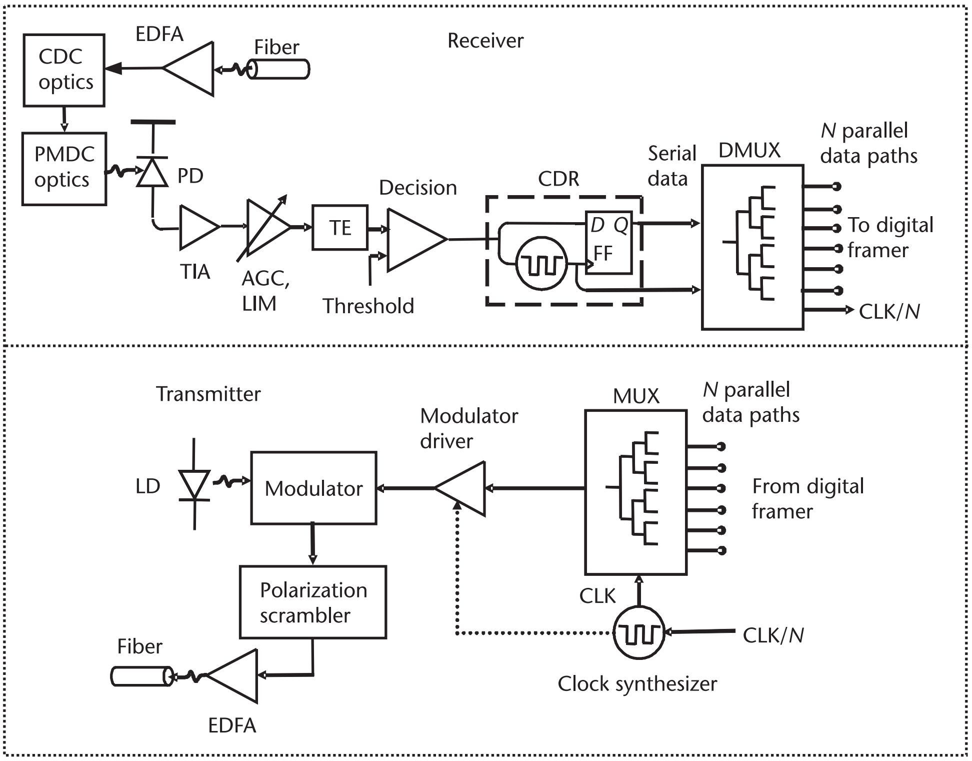


Figure 1.7 Block diagram of a fiber transceiver used in high-speed fiber networks.

modulation in fiber communications in some cases, the received data can be fed through a limiting or saturation amplifier without causing a bit error. In addition to optical dispersion compensation, a transversal equalizer can be inserted into the receiver path. A transversal equalizer can be a finite-impulse response (FIR) filter, which can provide additional compensation for both chromatic dispersion and PMD in the electronic domain. A decision feedback equalizer (DFE) can also be used to further equalize the distorted data. The DFE can be an infinite-impulse response filter, in which case stability issues need to be considered. At this stage, the signal is compared to a threshold level, and a decision is made as to whether a zero or a one has been received.

After the decision circuit, the digital data stream is fed into the clock and data recovery (CDR) circuit. Using the received data as a reference, the CDR circuit consists of a frequency synthesizer that generates a clock synchronized with the received data at the data rate. The received data is clocked by the recovered clock signal. This helps to square up the waveform and to remove intersymbol interference. The data and the recovered clock are then passed to the demultiplexer, which takes the serial data and converts it into *N* bits of parallel data at a frequency that is *N* times lower and sends the information to the digital framer. The digital framer then descrambles the bits to determine the information that was sent.

The transmission of data works much the same way, except in reverse. The digital framer in this case prepares the parallel data for transmission and passes it to a multiplexer (MUX) circuit, which generates a relatively high-transmission-rate data stream. A clock at the parallel data rate is also supplied. This clock is multiplied up *N* times by the clock synthesizer and used to time the data to be transmitted from the MUX to the laser driver or modulator driver circuit. The external modulator, in combination with a laser diode light source, can provide a high-speed, modulated, optical signal with reduced chirp. A polarization scrambler can be inserted after the optical modulator to randomize the polarization direction of the transmitted light. When light intensities along the principal polarization directions are the same, the system suffers the maximum power penalty due to PMD. By scrambling the polarization of the transmitted light, the PMD-related power penalty is reduced, on average, because there is a lower probability that the system will be locked in states with large power penalties. The optical signal is then amplified, typically with an EDFA, which is inserted after the scrambler to provide sufficient power for transmission of the optical signal along, for example, a standard single mode fiber (SSMF).

To increase the bandwidth over existing fiber optic backbones, multiple optical beams at different wavelengths can be launched simultaneously into the same optical fiber to form so-called dense wavelength division multiplexing (DWDM). The challenge of a DWDM system is mainly in the design of optical components. A fiber transponder with the above discussed fiber transceiver is needed for each wavelength in a DWDM system. Higher bandwidth can also be achieved by increasing the transmission data rate in the time domain, which challenges the fiber transceiver electronic design since the modulator driver, receiver front end, and CDR are required to operate at very high speed, such as 40 Gbps for OC-768 fiber communication systems.

In summary, synthesizer design for optical systems includes generation of the transmitting clock signal and the design of the CDR circuit. Requirements in the clock generation typically consist of the accuracy of frequency and the allowed amounts of frequency drift and jitter. In the CDR circuit, the main difference between this synthesizer and that for local oscillator generation in a radio system is that the input is not a crystal oscillator but the incoming data stream running at full frequency. Thus, the phase detector must also operate at full frequency and be able to cope with potentially long strings of zeros or ones. Thus, special phase-detector designs are required, and these will be discussed further in Chapter 6.

1.5 Frequency Synthesis for Modulation and Waveform Generation

As discussed in Section 1.2, frequency synthesizers are used to generate carrier frequencies in communication systems. The information to be transmitted or received is modulated onto the carrier frequency by various modulation schemes. Modulation of the baseband signal onto the RF carrier can be implemented in the following ways:

1. *Modulation in the analog domain:* The baseband signal is mixed with the RF carrier through an RF mixer, as illustrated in Figure 1.1, where both RF and IF mixers are implemented in the analog domain and the data converters need only operate at the baseband rate. The quadrature modulation is implemented at IF rather than RF, because the quadrature LO frequency can be generated relatively easily at a lower frequency with lower power consumption.
2. *Modulation in the digital domain:* If the IF is sufficiently low such that the digital-to-analog converter (DAC) and analog-to-digital converter (ADC) can convert the IF-modulated signal within their maximum sampling rate, the baseband modulation can also be done in the digital domain, as illustrated in Figure 1.8. In the digital domain, the channel filtering is implemented using digital FIR filters, and the quadrature IF carrier frequency is generated by a direct digital synthesizer (DDS). The baseband signal is modulated onto a low-speed carrier using multipliers in the digital domain, and the modulated signal is upconverted to the RF band using RF mixers in the analog domain. Implementing part of the transceiver functions in the digital domain reduces the complexity and the cost of the analog transceiver chip. However, the blocks that can be moved to the digital domain are limited by the converter-sampling rate. Converters with high resolution and a high sampling rate also have high power consumption and occupy a large amount of silicon area. Converters beyond the gigahertz sampling rate are usually implemented in advanced semiconductor technologies such as SiGe, which is not compatible with baseband CMOS chips.
3. *Direct modulation using a synthesizer:* In addition to the above-mentioned conventional modulation schemes, modulation can also be implemented directly through the frequency synthesizer. Frequency synthesizers are programmable. That is, they have an input that is used to control the

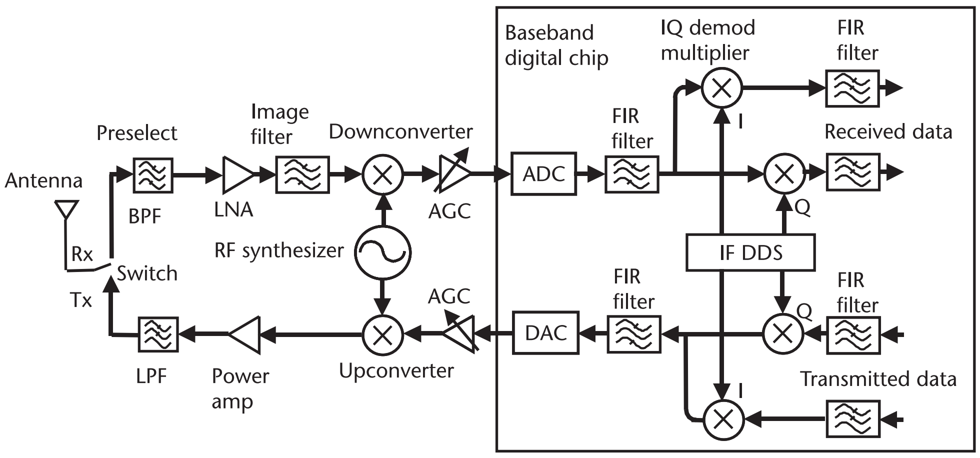


Figure 1.8 IQ modulation in a baseband digital chip.

frequency. Thus, if a modulation scheme is chosen that only has phase or frequency encoded data, this data can be added to the control signal, thereby directly modulating the synthesizer with data. In the receive path, the synthesizer is used to generate the carrier frequency, and a mixer is used to demodulate the received signal to baseband data. Figure 1.9 shows this direct-modulation scheme through a frequency synthesizer. In the transmit direction, the baseband signal representing the desired frequency variation of the output carrier directly modulates the carrier frequency. The direct-modulation scheme operates in the digital domain and, hence, simplifies the transmitter architecture as it avoids the need for analog mixers. However, its drawback is that the modulation bandwidth is limited by the synthesizer bandwidth. Recent researchers proposed to use predistortion schemes that apply an inverse transfer function of the phase-locked loop (PLL) to the baseband signal before it modulates the carrier frequency word. Thus, direct modulation would be less limited by the small synthesizer bandwidth. In order to achieve sufficient resolution for the baseband data, direct modulation normally requires more complexity in the frequency-controlling circuitry (as will be discussed in Chapter 9).

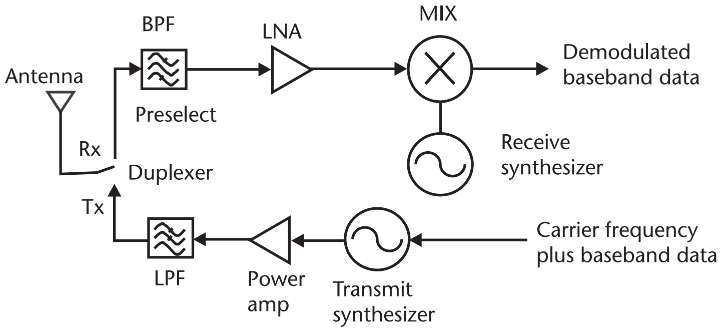


Figure 1.9 Direct modulation through a frequency synthesizer.

4. *Additional applications of waveform generation:* In addition to direct modulation, various waveforms can also be generated using synthesizers. An application for which this is highly suitable is radar. Radar operational requirements include ever-increasing demands for affordable, low noise signal and waveform generation. Low phase noise frequency generation is important in radar and transceiver designs since it increases the system signal-to-noise ratio, thus increasing the transmission range with the same transmission power, and it decreases the system bit-error rate, effectively increasing the data-transmission rate. The following will show some examples of waveform generation.
   - *Multicarrier generation:* Radar sensitivity can be greatly increased if the signals are transmitted simultaneously through multiple carriers. In order to distinguish the received signals, the carriers must differ from each other by frequency, phase, or the shape of the waveforms. A multifrequency transmission scheme is not favored since it increases transmission bandwidth and transceiver complexity. In contrast, multiphase or multiwaveform transmission is highly desirable since there is no bandwidth penalty.
   - *Complex waveform generation:* Another application of waveform generation is to produce various modulation schemes that are desired for novel transmitter architectures. Digitally generating highly complex, wide bandwidth waveforms at the highest possible frequency instead of near baseband would considerably reduce the transmitter architecture in terms of size, weight, power requirements, and cost. These waveforms are used for high-range resolution radars in sorting targets from clutter and low-probability-of-intercept communication applications. For example, a digital synthesis approach operating at carrier frequencies of greater than 10 GHz and bandwidths of greater than 1 GHz would greatly reduce transmitter complexity while improving the opportunity to pursue more multipurpose RF sensors.
   - *Modulated signal generation:* In addition to frequency synthesis, DDSs can also implement various modulation waveforms such as chirp, ramp, step frequency, minimum shift keying (MSK), phase modulation (PM), amplitude modulation (AM), quadrature amplitude modulation (QAM), and other hybrid modulations. Thus, they provide a low-cost digital approach to frequency, phase, and amplitude modulation, eliminating costly analog modulators associated with communication transceivers. The ability to generate a complex modulated waveform is a unique feature of the DDS approach.

1.6 Overview

The rest of this book will be devoted to discussing the details of frequency synthesizer design using IC technology. First, the system-level overview and "big picture" will be given, followed by the circuit details.

In Chapter 2, some synthesizer architectures will be discussed. In Chapter 3, details of system-level design of PLL-based frequency synthesizers will be described.

Since synthesizer design involves not only RF and analog design but also a significant amount of digital design, Chapter 4 will contain a summary of digital design techniques and issues. Subsequent chapters will contain the design details of the individual, fully integrated circuits that make up the PLL. In Chapter 5, continuing with the digital theme of Chapter 4, CMOS logic and current mode logic will be presented. In Chapter 6, specific applications of these digital circuits, such as dividers and phase detectors, will be discussed. Some of the dividers operate at the output frequency, requiring very high speed circuit techniques. Other dividers operate at lower frequencies where, typically, the main challenge is achieving low power dissipation. This will be followed in Chapter 7 by a discussion of the charge pump, a circuit with a digital input and an analog output. The output of the charge pump is connected to the loop filter, which will also be described in this chapter. In Chapter 8, the RF design of oscillators, the final loop component, will be presented. Topics will include inductor-capacitor (LC) oscillators and ring oscillators, and there will be some discussion of crystal oscillators, which are typically used as references for the PLL-based synthesizer. In Chapter 9, sigma-delta ($\Sigma\Delta$) modulators, which control the divider ratio in a PLL-based, fractional-N frequency synthesizer, will be discussed. Such design is largely digital in nature.

In addition to all these topics, the book will include a detailed discussion of the design of direct digital synthesizers in Chapter 10. These circuits have traditionally been used only in lower-frequency applications, but they are of growing importance as their speed and performance increases. Finally, Chapter 11 will contain a discussion of common digital modulations and direct modulation of frequency synthesizers.

There are also two appendices included at the end of the book. Appendix A is a review of control theory, and Appendix B is an overview of IC technology from a circuit designer's point of view. These have been included as review material separate from the main flow of the text as this material will be too rudimentary for advanced readers. Still, for readers less familiar with some of these topics, the authors' advice would be to read these two chapters first before proceeding with the rest of the text. As well, relevant control theory can be found in a number of general references [22–28].

References

[1] Rogers, J. W. M., and C. Plett, *Radio Frequency Integrated Circuit Design*, Norwood, MA: Artech House, 2003.

[2] Lee, T. H., *The Design of CMOS Radio Frequency Integrated Circuits*, 2nd ed., Cambridge, U.K.: Cambridge University Press, 2004.

[3] Razavi, B., *RF Microelectronics*, Upper Saddle River, NJ: Prentice Hall, 1998.

[4] Crols, J., and M. Steyaert, *CMOS Wireless Transceiver Design*, Dordrecht, the Netherlands: Kluwer Academic Publishers, 1997.

[5] Smith, J. R., *Modern Communication Circuits*, 2nd ed., New York: McGraw-Hill, 1998.

[6] Rappaport, T. S., *Wireless Communications: Principles and Practice*, 2nd ed., Upper Saddle River, NJ: Prentice Hall, 2001.

[7] Sklar, B., *Digital Communications: Fundamentals and Applications*, 2nd ed., Upper Saddle River, NJ: Prentice Hall, 2001.

[8] Couch, L. W., II, *Digital and Analog Communication Systems*, 6th ed., Upper Saddle River, NJ: Prentice Hall, 2001.

[9] Haykin, S., and M. Moher, *Modern Wireless Communications*, Upper Saddle River, NJ: Prentice Hall, 2004.

[10] Proakis, J. G., *Digital Communications*, 4th ed., New York: McGraw-Hill, 2000.

[11] Carson, R. S., *Radio Communications Concepts: Analog*, New York: John Wiley & Sons, 1990.

[12] Rohde, U. L., J. Whitaker, and A. Bateman, *Communications Receivers: DSP, Software Radios, and Design*, 3rd ed., New York: McGraw-Hill, 2000.

[13] Larson, L. E., (ed.), *RF and Microwave Circuit Design for Wireless Communications*, Norwood, MA: Artech House, 1997.

[14] Krauss, H. L., C. W. Bostian, and F. H. Raab, *Solid State Radio Engineering*, New York: John Wiley & Sons, 1980.

[15] Oklobdzija, V. G., et al., *Digital System Clocking: High Performance and Low Power Aspects*, New York: John Wiley & Sons, 2003.

[16] Wakerly, J. F., *Digital Design: Principles & Practices*, 3rd ed., Upper Saddle River, NJ: Prentice Hall, 2001.

[17] Bather, R. J., H. W. Li, and D. E. Boyce, *CMOS Circuit Design, Layout, and Simulation*, New York: IEEE Press, 1998.

[18] Martin, K., *Digital Integrated Circuit Design*, New York: Oxford University Press, 2000.

[19] Zhu, Q. K., *High-Speed Clock Network Design*, Norwell, MA: Kluwer Academic Publishers, 2003.

[20] Säckinger, E., *An Introduction to Broadband Circuits for Optical Communication*, Allentown, PA: Agere Systems Publication, 2001.

[21] Razavi, B., *Design of Integrated Circuits for Optical Communications*, New York: McGraw-Hill, 2002.

[22] Razavi, B., *Monolithic Phase-Locked Loops and Clock Recovery Circuits*, New York: Wiley-IEEE Press, 1996.

[23] Best, R. E., *Phase-Locked Loops: Theory, Design, and Applications*, 5th ed., New York: McGraw-Hill, 2003.

[24] Blanchard, A., *Phase-Locked Loops: Applications to Coherent Receiver Design*, New York: John Wiley & Sons, 1976.

[25] Gardner, F. M., *Phaselock Techniques*, New York: John Wiley & Sons, 1979.

[26] Egan, W. F., *Frequency Synthesis by Phase Lock*, New York: John Wiley & Sons, 2000.

[27] Wolaver, D. H., *Phase-Locked Loop Circuit Design*, Upper Saddle River, NJ: Prentice Hall, 1991.

[28] Crawford, J. A., *Frequency Synthesizer Design Handbook*, Norwood, MA: Artech House, 1994.

CHAPTER 2

Synthesizer Architectures

2.1 Introduction

In Chapter 1, applications that require synthesizers were introduced. This was done to provide background and motivation for the rest of this text, which is devoted to the details of making these circuits work. In this chapter, overviews of system-level configurations incorporating these circuits will be given. This will set the stage for the rest of the book, in which these systems will be examined in progressively more detail. Additional general information on synthesizers can be found in [1–21].

2.2 Integer-*N* PLL Synthesizers

An integer-*N* PLL is the simplest type of phase-locked loop synthesizer and is shown in Figure 2.1. Note that *N* refers to the divide-by-*N* block in the feedback of the PLL. The two choices are to divide by an integer (integer-*N*) or to divide by a fraction (fractional-*N*), essentially by switching between two or more integer values such that the effective divider ratio is a fraction. PLL-based synthesizers are among the most common ways to implement synthesizers, and this area is the subject of a great deal of research and development [22–54]. The PLL-based synthesizer is a feedback system that compares the phase of a reference f_r to the phase of a divided-down output of a controllable signal source f_{fb}, also known as a voltage-controlled oscillator (VCO). The summing block in the feedback is commonly called a phase detector. Through feedback, the loop forces the phase of the signal source to track the phase of the feedback signal; therefore, their frequencies must be equal. Thus, the output frequency, which is a multiple of the feedback signal, is given by

$$f_o = N \cdot f_{\text{ref}} \tag{2.1}$$

Due to divider implementation details, it is not easy to make a divider that divides by noninteger values. Thus, a PLL synthesizer of this type is called an integer-*N* frequency synthesizer. Circuits inside the feedback loop can be described by their transfer functions. These transfer functions can be designed to engineer the system dynamics to meet design specifications for the synthesizer. Typically, a lowpass filter (LPF) or lowpass network is the desired transfer function used in the loop. The details of the loop and the circuit components in the loop will be discussed in Chapters 3 through 8.

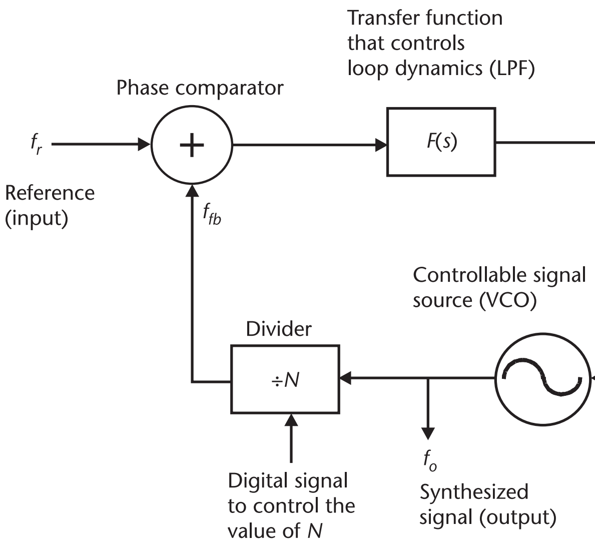


Figure 2.1 A simple integer-*N* frequency synthesizer.

Since N is an integer, the minimum step size of this synthesizer is equal to the reference frequency f_r. Therefore, in order to get a smaller step size, the reference frequency must be made smaller. This is often undesirable, so, instead, a fractional-N design is often used. This will be discussed next.

2.3 Fractional-*N* PLL Frequency Synthesizers

In contrast to an integer-N synthesizer, a fractional-N synthesizer allows the PLL to operate with high reference frequency while achieving a fine step size by constantly swapping the loop division ratio between integer numbers. As a result, the average division ratio is a fractional number. As will be shown in Chapter 3, a higher reference frequency leads to lower in-band phase noise and faster PLL transient response. In addition, for multiband applications, often the channel spacing of the different bands is skewed, requiring an even lower reference frequency if the synthesizer is to cover both bands.

Example 2.1: The Problem with Using an Integer-N Synthesizer for Multiband Applications

Determine the maximum reference frequency of an integer-N frequency synthesizer required to cover channels from 2,400 MHz to 2,499 MHz spaced 3 MHz apart, and channels from 5,100 MHz to 5,200 MHz spaced 4 MHz apart.

Solution: If an integer-N synthesizer were designed to service only one of these bands, then it would have a maximum reference frequency of 3 MHz in the first case and 4 MHz in the second case. However, if a synthesizer must be designed to cover both of these bands, then its step size must be 1 MHz to allow it to tune exactly to every frequency required.

In the simplest case, the fractional-*N* synthesizer generates a dynamic control signal that controls the divider, changing it between two integer numbers. By toggling between the two integer division ratios, a fractional division ratio can be achieved by time-averaging the divider output. As an example, if the control changes the division ratio between 8 and 9, and the divider divides by 8 for 9 cycles and by 9 for 1 cycle, and then the process repeats itself, then the average division ratio will be

$$\overline{N} = \frac{8 \times 9 + 9 \times 1}{10} = 8.1 \tag{2.2}$$

If the divider were set only to divide by 8, then it would produce 10 output pulses for 80 input pulses. However, now it will take 81 input pulses to produce 10 output pulses. In other words, the device swallows 1 extra input pulse to produce every 10 output pulses. In the PLL synthesizer, this time average is dealt with by the transfer function in the loop. This transfer function will always have a lowpass characteristic. Thus, it will deliver the average error signal to the VCO. As a result, the output frequency will be the reference frequency multiplied by the average division ratio. However, toggling the divider ratio between two values in a repeating manner generates a repeating time sequence. In the frequency domain, this periodic sequence will generate spurious components (or spurs) at integer multiples of the repetition rate of the time sequence. Such spurious components can be reduced by using $\Sigma\Delta$ modulators in which the division ratio is randomized. $\Sigma\Delta$ modulators will be discussed in Chapter 9.

2.3.1 Fractional-*N* Synthesizer with Dual-Modulus Prescaler

Figure 2.2 illustrates one way to implement a simple fractional-*N* frequency synthesizer with a dual-modulus prescaler $P/P+1$. Note that it is called a dual-modulus prescaler because it can be programmed to two division ratios. As discussed in the previous section, the fractionality can be achieved by toggling the divisor value between two values, P and $P+1$. The modulus control signal (C_{out}) is generated using an accumulator (also called an integrator or adder with feedback, or a counter) with size of F (or $\log_2 F$ bits). That is, an overflow occurs whenever the adder output becomes equal to or larger than F. At the ith clock rising edge, the accumulator's output y_i can be mathematically expressed as

$$y_i = (y_{i-1} + K_i) \bmod F \tag{2.3}$$

where y_{i-1} is the output on the previous rising clock edge, and K_i is a user-defined input, and the value of K_i will determine the fractional-divider value. Its use will be illustrated shortly in Example 2.2. Note that the modular operation ($A \bmod B$) returns the remainder of ($A \div B$) and is needed for modeling the accumulator overflow.

Example 2.2: A Simple Accumulator Simulation

Describe the operation of a 3-bit accumulator with input $K = 1$ and $K = 3$, assuming the accumulator seed value (i.e., the initial accumulator output value) is equal to zero.

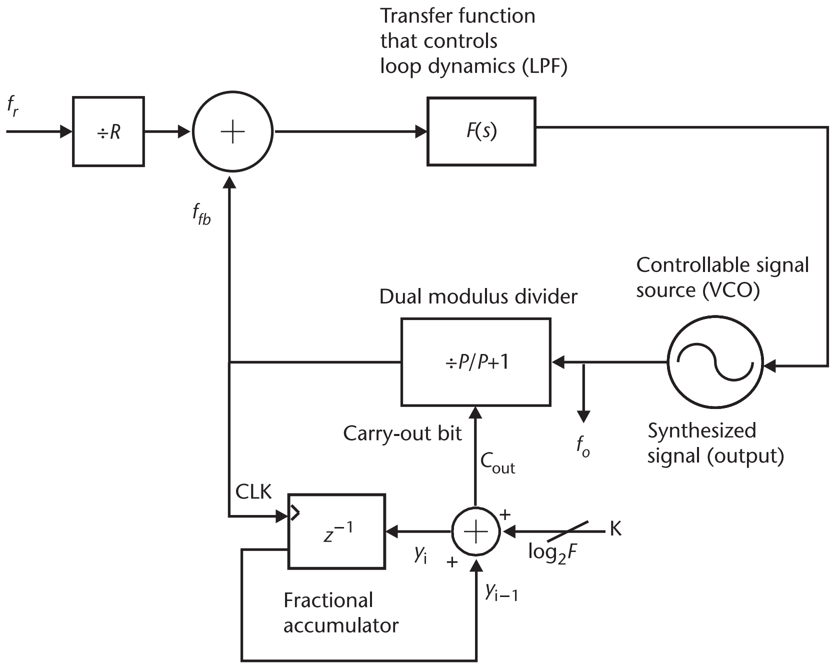


Figure 2.2 A fractional-*N* frequency synthesizer with a dual-modulus prescaler.

Solution: Note that Verilog notation for binary numbers will be used here. For instance, a zero in 3-bit binary format = 3'b000, where "3" represents the number of bits and "b" denotes the binary format. The detailed Verilog coding for digital designs will be introduced in Chapter 4. If the accumulator has 3 bits, the size of the accumulator is $2^3 = 8$, or $F = 8$, even though the largest value that can be stored is 3'b111, corresponding to 7. If input word $K = 1$, namely, 3'b001, the accumulator value y increases by 1 every cycle until it reaches the maximum value that can be represented using 3 bits, namely, $y_{max} = 7 =$ 3'b111. After this point, the accumulator will overflow, leaving its value $y = 0$ and $C_{out} = 1$. It will take eight clock cycles for the accumulator to overflow if $K = 1$. In other words, the accumulator size $F = 8$. For $K = 3 =$ 3'b011, the accumulator adds an increment value of 3 every cycle and, thus, overflows more often. The accumulator value and its carry out C_{out} are summarized cycle by cycle in Table 2.1 for $K = 1$ and Table 2.2 for $K = 3$.

As shown in the above example, for the $K = 1$ case, C_{out} is high for one cycle and low for seven cycles within every eight clock cycles, so the frequency of C_{out} is $f_{clk}/8$. For the $K = 3$ case, C_{out} is high for three cycles and low for five cycles

Table 2.1 Accumulator Operations with $F = 8$, $K = 1$

Clock Cycle i	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*	*16*	*17*	*18*
y_i	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2
y_{i-1}	N/A	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1
C_{out}	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0

Table 2.2 Accumulator Operations with $F = 8$, $K = 3$

Clock Cycle i	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*	*16*	*17*	*18*
y_i	0	3	6	1	4	7	2	5	0	3	6	1	4	7	2	5	0	3	6
y_{i-1}	N/A	0	3	6	1	4	7	2	5	0	3	6	1	4	7	2	5	0	3
C_{out}	1	0	0	1	0	0	1	0	1	0	0	1	0	0	1	0	1	0	0

within every eight clock cycles, so the frequency of C_{out} is $3f_{\text{clk}}/8$. In general, for a constant input word K, the accumulator carry-out will be high for K cycles and will be low for $F - K$ cycles. Also note that the frequency of C_{out} will be equal to

$$f_{C_{\text{out}}} = \frac{Kf_{\text{clk}}}{F} \tag{2.4}$$

If the dual-modulus prescaler divides by P when C_{out} is low and divides by $P + 1$ when C_{out} is high, the average VCO output frequency is

$$f_o = \frac{f_r}{R}\left[\frac{(P+1)K + P(F-K)}{F}\right] = \frac{f_r}{R}\left(P + \frac{K}{F}\right) \tag{2.5}$$

Because fractionality is achieved by using this accumulator, it is often called a fractional accumulator. It has a fixed size F due to the fixed number of accumulator bits built into the hardware. The dual-modulus prescaler ratio P is normally fixed as well. The only programmable parameter for the architecture shown in Figure 2.2 is the accumulator input K, which can be programmed from one to a maximum number of F. Thus, since K is an integer, from (2.5) it can be seen that the step size of this architecture is given by

$$\text{Step size} = \frac{f_r}{RF} \tag{2.6}$$

where R is normally fixed to avoid changing the comparison frequency at the input. Note that R is normally as small as possible to minimize the in-band phase noise contribution from the crystal. Thus, step size is inversely proportional to the number of bits ($\log_2 F$); as a result, the accumulator is normally used to reduce synthesizer step size without increasing R and degrading the in-band phase noise. Of course, we need to solve the problem with spurs associated with the fractional-N scheme, and this will be a major focus of Chapter 9.

2.3.2 An Accumulator with Programmable Size

For some applications, a programmable synthesizer step size may be needed. Therefore, it is often desirable to program the accumulator size F as well. Next, it will be explained how to program the size of an accumulator from one to a maximum value of F, with a maximum fixed number of bits available in the hardware.

Figure 2.3 illustrates the architecture for a fractional accumulator with size G, programmable from one to a maximum size of F. Note that the number of accumula-

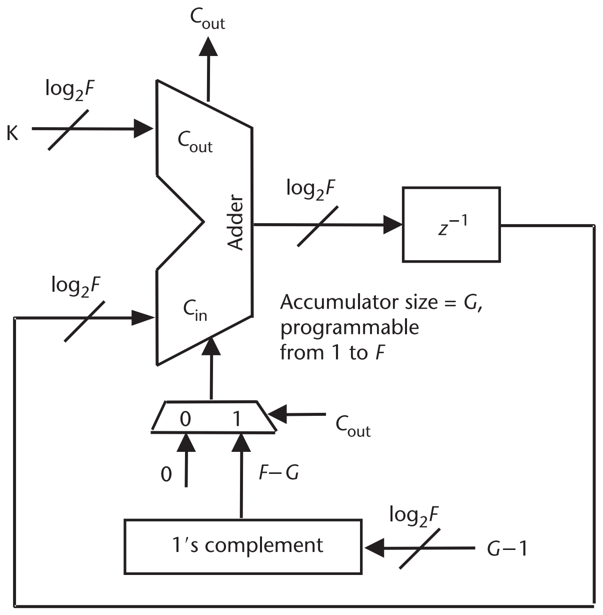


Figure 2.3 Fractional accumulator with size G programmable from 1 to a maximum size of F.

tor bits is $n = \log_2(F)$. In Verilog binary format, the maximum accumulator value is $F = \{n\{1\text{'b}0\}\}$, which is n concatenated zeros. In general, in modular F arithmetic, 0 and F are equivalent. For instance, a 3-bit accumulator has a maximum value $F = 8$, which is represented by three zeros (3'b000) in binary. Note that the maximum number that can be stored in the accumulator is $F - 1$, which is 3'b111 for the 3-bit accumulator. When the accumulator value reaches F or greater, a carry-out bit is generated. Compared to a simple accumulator without size programmability, the above accumulator has a carry-in C_{in} that adds $F - G$ to the accumulator whenever the carry-out C_{out} is high. Thus, the accumulator will always start at value $F - G$. If the input is $K = 1$, it will take G clock cycles for the accumulator to reach the overflow point F ($F - G + G = F$), which is equivalent to having an accumulator with size G.

Since F in modulo F arithmetic is equivalent to zero, $F - G = -G$. Note that $-G$ can be obtained by taking the two's complement of G, that is, by inverting all the bits of G and then adding 1 to the result. To avoid the addition for the two's complement operation, the input to the synthesizer can be $G' = G - 1$ instead of G. Thus, $F - G = -G$ can be obtained simply by inverting all the bits of the input G'. This operation will result in the one's complement of G'.

Example 2.3: Programmable Accumulator

Modify the 3-bit fractional accumulator designed in the previous example to have a size of 6.

Solution: Let the input be $G' = G - 1$ instead of G. For example, if $G' = G - 1 = 0 =$ 3'b000 is the input, the accumulator size is $G = 1$; if $G' = G - 1 = 7$ = 3'b111, then this corresponds to a size of $G = 8$, which is the maximum accumulator size using 3 bits. To program the accumulator size as 6, we input $G' = G - 1$

= 5 = 3'b101. The accumulator start value is the maximum size F (3'b000) minus the programmed size G, resulting in $-G$, which is the one's complement of G' (i.e., 3'b010), and is obtained by inverting every bit of G'. When the accumulator is initialized, the adder is loaded with 3'b010 = 2. With an input of $K = 1$ = 3'b001 (unit step), it will take six cycles from 3 = 3'b011 to 8 = 3'b000 to reach the overflow point. Thus, this implements an accumulator with size of 6. Table 2.3 summarizes the accumulator value cycle by cycle; clearly, the carry-out rate is now one-sixth of the clock cycle with $K = 1$.

Now programmability for both the numerator and denominator of the fractional part of the divider ratio K/F has been implemented. However, the integer part of the division ratio in (2.5) is a fixed number P, which limits the synthesizer output to a frequency range from $f_r P/R$ to $f_r(P + 1)/R$. It is often desirable to program the synthesizer over a wider frequency range with programmability for the integer part of the division ratio as well. For instance, a multiband WLAN synthesizer needs to synthesize channel frequencies at the 2.4 GHz and 5.3 GHz bands with the same reference frequency. To build a fully programmable synthesizer, a multimodulus divider (MMD) can be used, as is discussed in the following section.

2.3.3 Fractional-*N* Synthesizer with Multimodulus Divider

Replacing the dual-modulus divider with an MMD, the synthesizer architectures shown in Figure 2.2 can be modified to a more generic form, as illustrated in Figure 2.4. Using an MMD has the advantage that the range of frequencies over which the synthesizer can be tuned is expanded, compared to the previous architecture. The synthesizer output frequency is given by

$$f_o = \frac{f_r}{R}\left(I + \frac{K}{F}\right) \tag{2.7}$$

where I is the integer portion of the loop divisor, and, depending on the complexity of the design, I could have many possible integer values. For instance, if a loop division ratio of 100.25 is needed, we can program $I = 100$, $K = 1$, and $F = 4$. The MMD division ratio is toggled between 100 and 101.

A popular MMD topology using cascaded 2/3 cells will be discussed in Chapter 6. With an n-bit modulus control signal, the MMD division ratio is given by

$$N_{\mathrm{MMD}} = P_1 + 2^1 P_2 + \ldots + 2^{n-2} P_{n-1} + 2^{n-1} P_n + 2^n \tag{2.8}$$

Table 2.3 Accumulator Operations with $F = 8$, $G = 6$, and $K = 1$

Clock Cycle i	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*	*16*	*17*	*18*
y_i	2	3	4	5	6	7	2	3	4	5	6	7	2	3	4	5	6	7	2
y_{i-1}	N/A	2	3	4	5	6	7	2	3	4	5	6	7	2	3	4	5	6	7
C_{out}	1	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	1

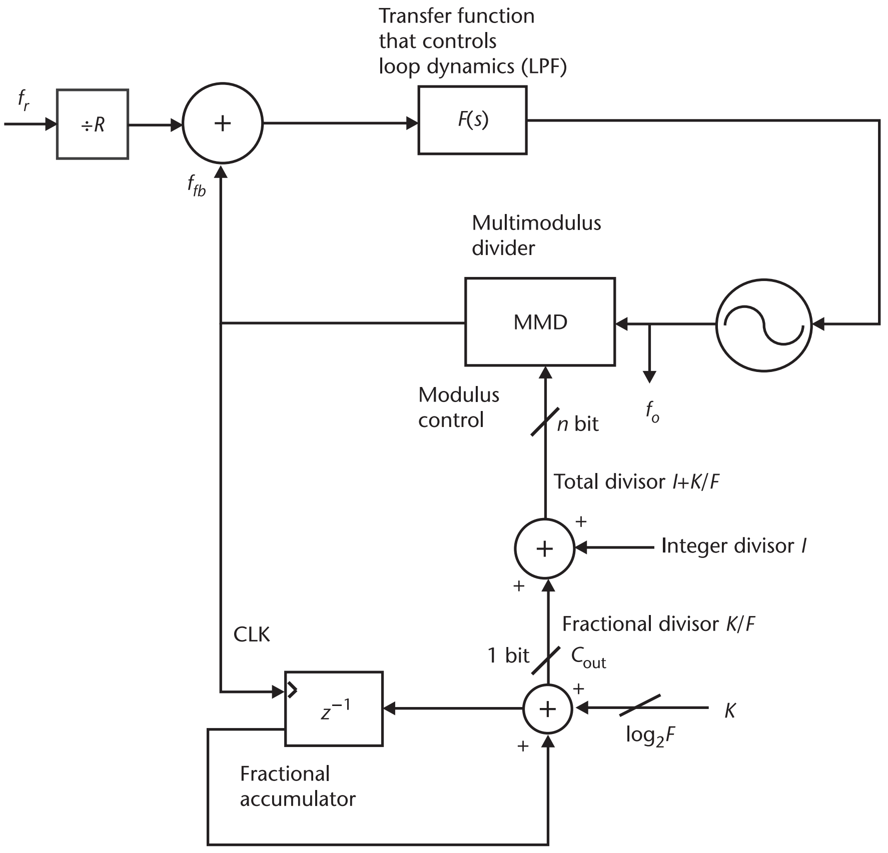


Figure 2.4 A fractional-*N* frequency synthesizer with an MMD.

where the MMD programming range is 2^n to $2^{n+1} - 1$. For instance, a 6-bit MMD can be programmed from 64 to 127. The MMD program range can be further extended through the use of additional logic [55]. Wide programming range is critical for multiband frequency synthesis, especially when ΣΔ noise shaping is employed, as discussed in Chapter 9.

2.3.4 Fractional-*N* Spurious Components

The above-discussed fractional-*N* architectures suffer from a common side effect of generating spurious components associated with periodically toggling the loop division ratio. Recall that any repeated pattern in the time domain causes spurious tones in the frequency domain. The fractional accumulator periodically generates the carry-out that toggles the loop division ratio. It is expected that there should be spurious tones at multiples of the carry-out frequency $(f_r/R) \cdot (K/F)$. In the following example, fractional-*N* spurs are analyzed with simulation and measurement.

Example 2.4: The Use of Accumulators in Fractional-N Synthesizers
Design a fractional-*N* synthesizer architecture for synthesizing 11 channels from 819.2 MHz to 820.96 MHz with a step size of 160 kHz and reference comparison

frequency of $f_r/R = 5.12$ MHz. Determine the frequencies of fractional-*N* spurious components.

Solution: The synthesizer step size is given by $\frac{f_r}{R} \cdot \frac{1}{F} = 160$ kHz. Since the comparison frequency is $f_r/R = 5.12$ MHz, the fractional accumulator size can be chosen as

$$F = \frac{f_r}{R} \cdot \frac{1}{160 \text{ kHz}} = \frac{5{,}120 \text{ kHz}}{160 \text{ kHz}} = 32$$

which can be implemented using a 5-bit accumulator. The accumulator input (i.e., the fine-tune frequency word K) can be programmed from 0 to 10 to cover the 11 channels from 819.2 MHz to 820.96 MHz with a step size of 160 kHz (the first channel does not require any fractionality). The integer divisor ratio (i.e., the coarse-tune frequency word I) can be determined by the channel frequency. For instance, the first channel frequency is synthesized as

$$\frac{f_r}{R}\left(I + \frac{0}{F}\right) = \frac{f_r}{R} \cdot I = 819.2 \text{ MHz}$$

which leads to $I = 160$. Hence, the loop total divisor is given by $N = 160 + K/32$, where $K = 0, 1, \ldots 10$. If a dual-modulus divider is used to construct a fractional-*N* synthesizer, as illustrated in Figure 2.2, the dual-modulus divider ratio should be $P/P + 1 = 160/161$, which is not the best solution as far as power consumption and circuit speed are concerned. There are better circuit implementations, such as using an MMD or a pulse-swallow divider, which allows much of the implemented circuitry to operate at much lower speeds. Details of such dividers will be discussed in Chapter 6. If a fractional-*N* architecture with MMD is used, as illustrated in Figure 2.4, a 7-bit MMD with a programmable range from 128 to 255 is needed based on (2.8). In any of the above solutions, the loop divisor of the fractional-*N* architecture is toggled between 160 and 161. The modulus control is generated by the accumulator carry-out, which has a frequency of $(f_r/R) \cdot (K/F)$. Thus, the loop is divided by 160 for K reference cycles and divided by 161 for $F - K$ cycles, which results in an average division ratio of $160 + K/F$. As an example, for the second channel with $K = 1$ and $F = 32$, the simulated fractional accumulator output is given in Figure 2.5. As shown, the fractional accumulator outputs a carry-out in every 32 comparison cycles, which forces the loop divider to divide by 160 for 31 cycles and to divide by 161 for 1 cycle, as shown in Figure 2.5(a). The periodic phase correction pulse due to dividing by 161 generates fractional spurs with uniform spacing of $f_r/R/32 = 160$ kHz, as shown in Figure 2.5(b).

The spurious tones generated by the accumulator will appear in the output spectrum of the synthesizer. Figure 2.6 presents a measured spectrum at a fractional-*N* synthesizer output with loop divisor $N = 160 + 1/32$ and the comparison frequency $f_r/R = 5.12$ MHz. Fractional spurs at integer multiples of $\frac{f_r}{R} \cdot \frac{K}{F} = 160$ kHz are observed. The roll-off of the spur magnitude as frequency increases is due to the loop lowpass roll-off characteristics.

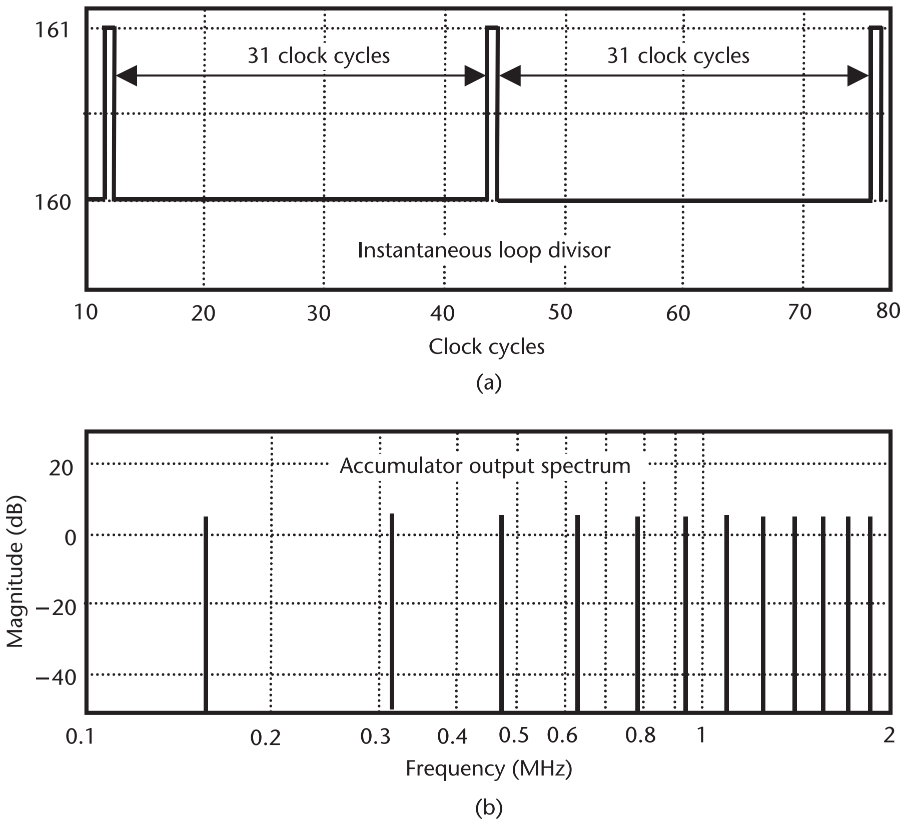


Figure 2.5 Simulated fractional accumulator output with loop divisor $N = 160 + 1/32$ and the comparison frequency $f_r/R = 5.12$ MHz: (a) time domain, and (b) frequency domain.

A fractional-N synthesizer achieves fine step size and low in-band phase noise with the penalty of fractional spurious tones, which comes from the periodic division ratio variation. Fractional spurs may be removable by using a high-order loop transfer function if the closest spur is outside of the PLL bandwidth. Note that the spacing of the closest spur to the carrier is determined by the synthesizer step size. For a synthesizer with a fine step size smaller than the transfer function bandwidth, it is thus practically impossible to remove fractional spurs by using a loop LPF. Reducing the loop bandwidth to combat the fractional spurs means that you have to pay the penalty of longer lock time and increased out-of-band phase noise due to the VCO, as will be discussed in Chapter 3. Even if the closest spur is outside of the loop filter bandwidth, removing those spurs normally requires a high-order loop filter with sharp roll-off, which increases the complexity and cost of the synthesizer. To remove the fractional spurious components for a synthesizer with fine step size, the best solution is to employ a ΣΔ noise shaper in the fractional accumulator; these circuits will be a topic of Chapter 9. Their function is to break up the repeated patterns of the loop divisor time sequence without affecting its average division ratio. This will result in the reduction or elimination of the spurs in the spectrum.

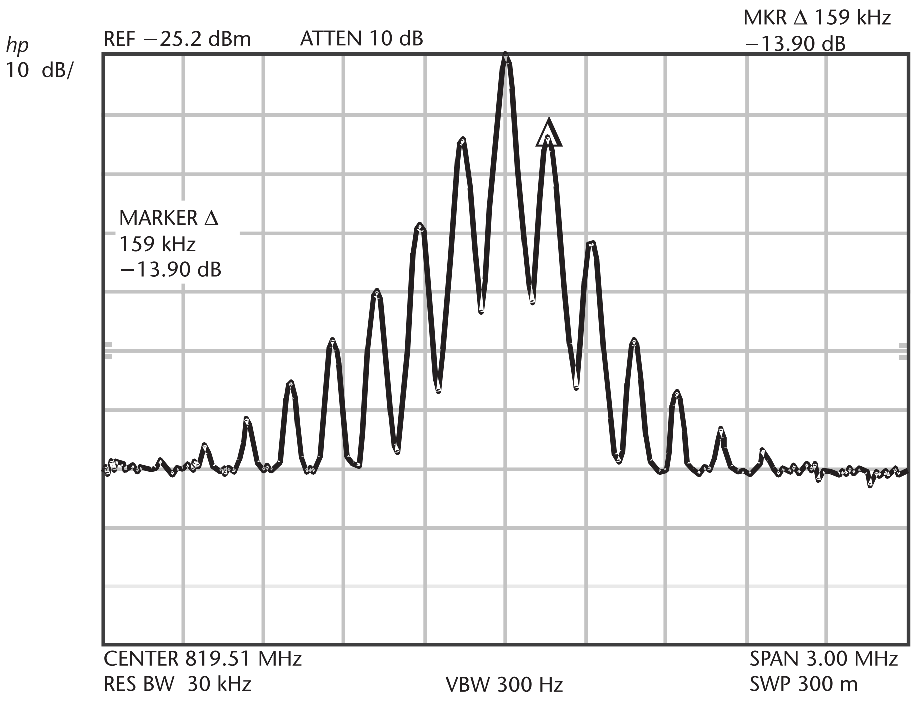


Figure 2.6 Measured output spectrum of a fractional-N frequency synthesizer with loop divisor $N = 160 + 1/32$ and the comparison frequency $f_r/R = 5.12$ MHz.

2.4 Delay-Locked Loops

Recently, the delay-locked loop (DLL) has become more popular due to its simplicity and inherent performance advantages [56–61]. A DLL is similar to a PLL except that the VCO is replaced with a voltage-controlled delay line (VCDL), as shown in Figure 2.7. The idea is that the input and output of the delay line are sent to the phase detector. The output of the phase detector, representing the phase error, is amplified and used to control the delay in the delay line. Thus, by feedback, the delay in the delay line is adjusted to equal the period of the reference or the data input. The DLL may also lock to a multiple of the reference period, which is not

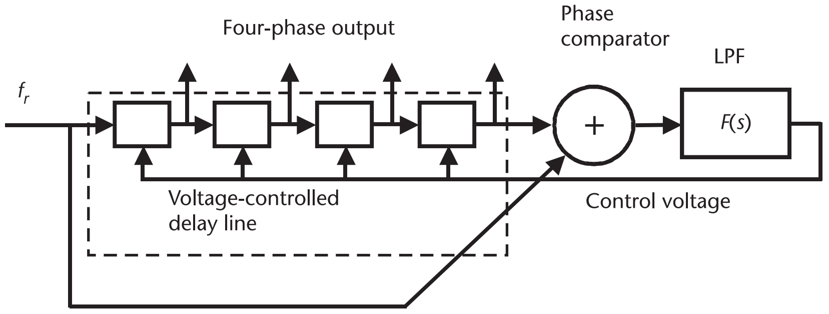


Figure 2.7 A DLL.

desirable if precise delay between clock edges is required. As an application example, if there are four equal delay components in the delay line and if the input and the final output are both aligned with the incoming clock, then each consecutive intermediate output has a 90° phase shift. In other words, this is a multiphase (four phases in this case) clock generator. The delay line of a DLL can comprise delay elements potentially equivalent to delay elements that could be used as part of a voltage-controlled ring oscillator in a PLL. In both the VCDL and the ring oscillator, jitter in one stage is passed on to the following stages. Thus, the jitter accumulates. In the VCO, this jitter is fed back to the first stage, while in the VCDL, the first stage always gets a fresh start. For this reason, the DLL will be inherently less noisy than the PLL.

In terms of mathematics, in a PLL, the VCO output phase is proportional to the integral of the input control voltage. However, a delay line has an output delay (or phase shift) proportional to the control voltage of an integrator. Thus, a DLL has a transfer function inherently one order lower than that of the PLL. This means it may be easier to stabilize than the PLL. However, it also means that unless there is another component in the loop with high or infinite gain, there will be a finite phase error; that is, the output edges from the delay line will not be exactly lined up with the inputs. The solution to this is to use a loop filter with infinite gain. Such a filter, which can be realized with a charge pump, will be further discussed in Chapter 7.

A common use of the DLL is for clock recovery. A typical circuit is shown in Figure 2.8. In this case, the data input is compared to the clock, using a special phase detector. This phase detector, which is commonly used in data-recovery applications, produces an output only when there are input edges. Examples of such phase detectors will be discussed further in Chapter 6. In this circuit, the data at the output of the delay line is compared with the clock. Any phase error is fed back to adjust the delay in the delay line to bring the data into alignment with the clock. We note that without a VCO, the clock must inherently be at the same frequency as the data.

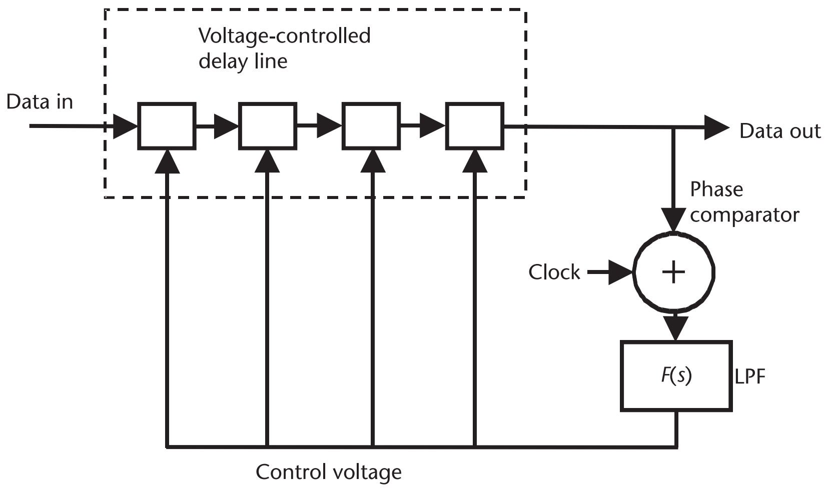


Figure 2.8 Use of a DLL for CDR applications.

2.5 Clock and Data Recovery (CDR) PLLs

Optical receivers, discussed in Chapter 1, require synthesizers as well. In this application, the synthesizer is required to recover a clock signal from the incoming data. After the decision stage in the optical receiver (see Chapter 1), the data stream can be used to recover the clock, which is synchronized with the data. An example CDR circuit, as shown in Figure 2.9, is essentially a PLL similar to the PLL-based synthesizer that has just been considered. The main difference is that a PLL for a conventional frequency synthesizer has a clean reference signal typically at a much lower frequency compared to the carrier frequency as an input. The PLL for a CDR has a potentially noisy and attenuated random-bit stream at high frequency as an input. Thus, the phase detector is specifically designed for the expected input bit stream.

As shown in Figure 2.9, a typical phase detector could consist of four *D*-type flip-flops that are clocked by quadrature phase clock signals with 90° of phase separation. The quadrature phase clocks can operate at half of the input data rate. The data stream can be sampled at equally spaced time points. At the locked state (i.e., when the recovered clock is synchronized to the input data), the *D*-flip-flops with 0° and 180° of phase shift sample the data stream in the middle of the eye patterns (in the middle of a bit transmission), and the *D*-flip-flops with 90° and 270° of phase shift sample the data stream at the edge of the eye pattern (at the end and beginning of a bit transmission). Phase-detect logic circuitry compares the binary phase detector outputs and generates a control signal, filtered by the loop's transfer function or lowpass filter, which is used to drive the VCO. If, for any reason, the clock edges produced by the VCO start to drift out of phase with the

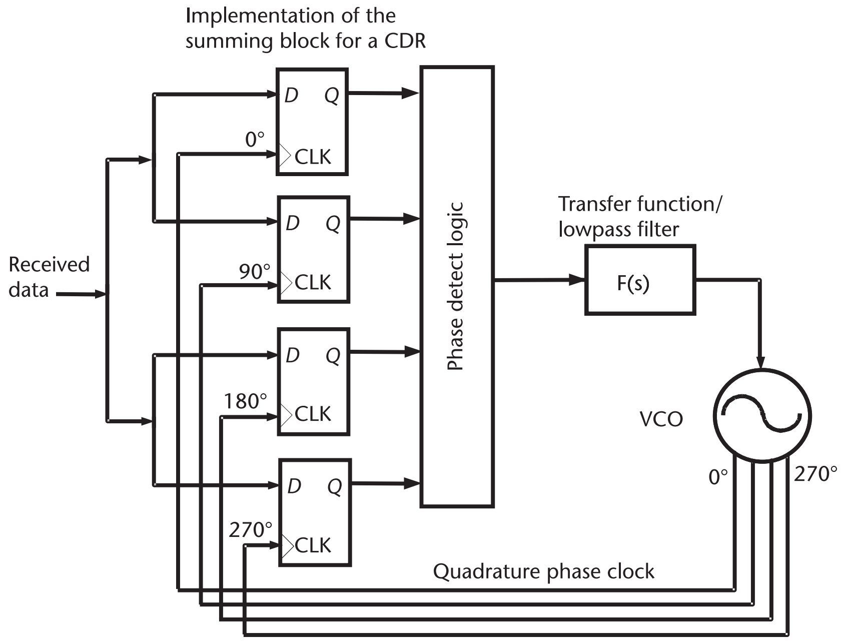


Figure 2.9 A CDR PLL with a quadrature phase detector operating at half of the data rate.

data, this circuitry will apply feedback to correct the VCO's phase. Such circuitry will be discussed in more detail in Chapter 6. It will be shown that the lowpass-loop filter determines the PLL performance and, thus, affects the CDR jitter performance. The VCO can be a ring oscillator because such oscillators can intrinsically generate quadrature phase clocks, as will be discussed in Chapter 8. In addition, as mentioned above, the four-phase VCO can operate at half of the data rate, which is of benefit to high-speed circuit design for low phase noise. The recovered data can then be retimed and demultiplexed into a lower data rate by a 1:*N* demultiplexer. In addition, the recovered clock signal is divided into a relatively low frequency, which can be used to drive the baseband digital IC. To ensure error-free data capture at the input of the baseband digital IC, the received data itself needs to be retimed in the CDR using the recovered clock. Normally, data retiming is implemented in the phase detector used to form the CDR PLL. In the CDR design, the output clock jitter performance is also rather important. Ideally, it should sample in the middle of the bit period when the voltage is changing the least, thereby minimizing the chance that there will be an error in determining the value of the bit. However, if the clock experiences jitter, then sampling will not occur at this optimal point, and if the jitter becomes too large, then an error is likely. Thus, a low jitter clock is required.

The main challenge with the optical transmitters in regard to frequency synthesis is in the CDR circuit. This circuit must recover the clock from the data. There are two typical ways to transmit the data. Either return-to-zero (RZ) or nonreturn-to-zero (NRZ) is used, as illustrated in Figure 2.10. In NRZ format, each bit is exactly one bit period wide. Thus, long strings of ones or zeros could result in the absence of voltage transitions, which could affect phase-detector design. Also note that, even if a stream of alternating ones and zeros were transmitted, this would be a square wave with a frequency equal to one-half of the bit rate. Thus, there is actually no power in the data stream at the frequency at which a clock needs to be generated. This makes clock recovery very difficult, and, usually, a circuit that takes the derivative of the incoming bit stream must be used. This new waveform will have power at the bit frequency and can be used to help synthesize the clock. The second way of transmitting data, RZ, uses shorter duty-cycle pulses so that, in this case, there is actually energy in the waveform at the clock frequency. This is much more convenient for the CDR circuit but uses more fiber bandwidth than the NRZ format. In summary, from the electronic design point of view, RZ data format benefits CDR design due to the presence of a distinct clock frequency tone in the received data, while NRZ data format benefits the design of the input high-

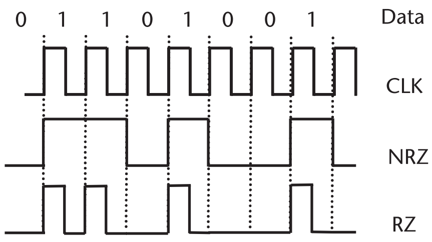


Figure 2.10 Data stream in NRZ and RZ format.

frequency circuitry due to its lower bandwidth. RZ has 2-dB better sensitivity due to its higher peak optical intensity for the same average power compared to NRZ format. Since NRZ-based CDR has been well developed, the NRZ data format is often adopted for high-speed applications such as OC-768 fiber networks due to its lower bandwidth requirement.

2.6 Direct Digital Synthesizers

The PLL synthesizer is by far the most popular frequency-synthesis technique and has been widely used in communication systems and radar applications. The PLL synthesizer is capable of generating high frequencies with low phase noise. However, it is expensive due to the use of analog components, and it has a long settling time due to the narrow bandwidth of the loop filter. It has small tuning range due to the limited tuning range of the VCO and difficulty achieving fine step size while minimizing fractional spurious components. The PLL synthesizer is also not very suitable to perform various modulations, although direct modulation through a PLL is possible with predistortion circuitry.

In comparison to PLL synthesis, direct digital synthesizer (DDS) uses digital circuits to create, manipulate, and modulate a signal digitally and eventually to convert the digital signal to its analog form using a DAC. DDS provides many advantages, including fast frequency switching, fine frequency-tuning resolution, and continuous-phase switching, and it allows direct phase and frequency modulations in the digital domain. Moreover, DDS is built on inexpensive and reliable digital CMOS technology and is very suitable for embedding in baseband CMOS chips. Due to the enormous revolution of digital technology, DDS has become a frequency-synthesis technique of growing importance in applications such as test instruments, radars, medical imaging, satellite, and wireless communications [62–106].

The DDS concept was described by Manessewitz as early as 1980 [17], but its implementation did not become feasible until the 1990s when IC technology became sufficiently advanced. DDS employs the waveform digital-synthesis method. A sine wave, triangular wave, or square wave is computed as a series of samples. These samples are converted to an analog output waveform through a DAC that has to be clocked with a very precise, low-jitter clock waveform. Low jitter is equivalent to having low phase noise. Since there are no feedback loops in the DDS, the output frequency can be programmed to change instantaneously whenever a new frequency word is loaded. With suitable design of the digital component that generates the sample values, phase coherence suitable for low spurious, fast frequency hopping can also be obtained. There are, however, two major deficiencies in current DDS synthesizers because IC technology is still not sufficiently advanced. First, the DDS suffers from a high level of spurious output signals due to limitations of the DAC. The spurious level can be reduced by clocking the DDS at a much higher frequency than the output frequency and by using $\Sigma\Delta$ noise-shaping techniques. A second deficiency is that, as the sampling frequency increases, the power required both for the digital waveform computing circuitry and the output DAC increases approximately in proportion.

2.6.1 Direct Digital Synthesizer with Read-Only Memory Lookup Table

A conventional DDS architecture with read-only memory (ROM) lookup table is shown in Figure 2.11. It utilizes a phase accumulator to generate the phase word based upon the desired input frequency control word (FCW), which is stored in the frequency register. The frequency word is continuously accumulated in the phase accumulator using an N bit adder. This means that the FCW is continuously added to the previous content of the phase register. The output of the adder is sampled at the reference sample clock rate (DDS f_{clk} in Figure 2.11) by an N bit register. When the accumulator reaches the N bit maximum value, the accumulator rolls over and continues. The DDS output frequency is thus given by

$$f_{\mathrm{out}} = \frac{\mathrm{FCW}}{2^N} f_{\mathrm{clk}} \tag{2.9}$$

The sampled output of the phase accumulator is then used to address a ROM lookup table that stores the sinusoidal magnitude values. The ROM lookup table hence converts the phase word into sine/cosine amplitude words. The binary digital amplitude word is further used to drive a binary weighted DAC. The DAC output is a sample-and-hold circuit that takes the digital amplitude words, converts the value into an analog voltage, and holds the value for one sample clock period. Since the DAC output is sampled at the reference clock frequency, a waveform smoothing LPF is typically used to eliminate alias components.

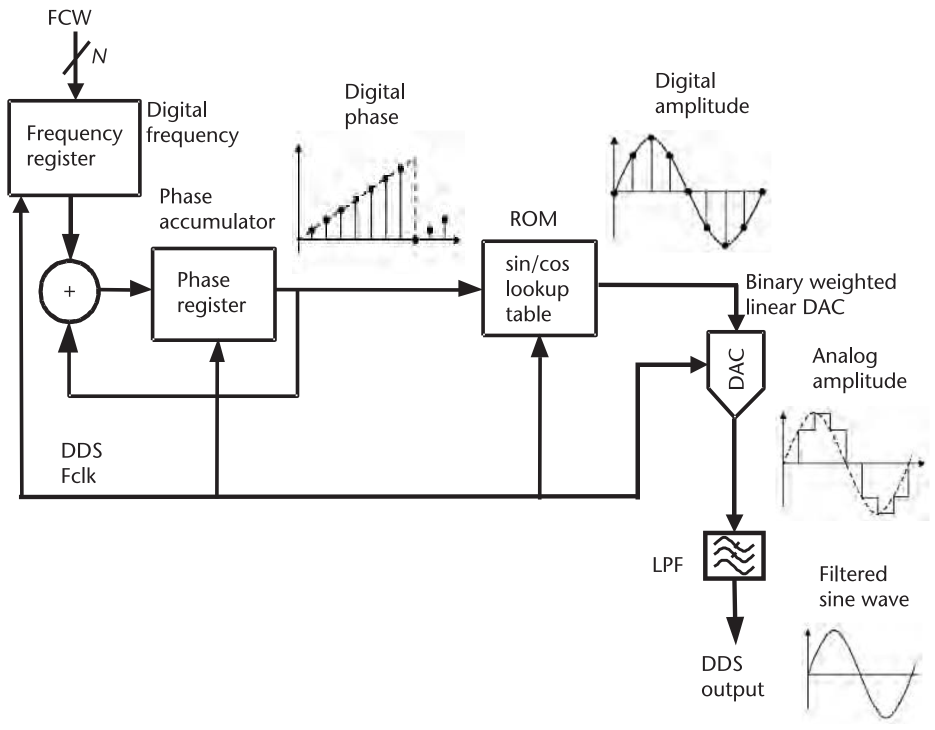


Figure 2.11 Direct digital synthesis with ROM lookup table.

It is also easy to see that this synthesizer can produce a modulated signal by changing the FCW. If this input is not static but rather changes with time, this circuit can directly produce a frequency modulated (FM) signal.

2.6.2 ROM-Less Direct Digital Synthesizer

Direct digital synthesis provides precise frequency resolution and direct-modulation capability. However, the majority of the DDS circuits designed so far are limited to low-frequency applications with clock frequencies less than a few hundred megahertz, and the output frequencies must be even less than this. Digitally generating highly complex, wide-bandwidth waveforms at the highest possible frequency instead of down near baseband would considerably reduce the weight, power consumption, size, and cost of a transmitter architecture. As shown in Figure 2.11, the conventional DDS architecture utilizes a ROM lookup table to convert the accumulated phase word into sine/cosine words that are further used to drive a linear, binary-weighted DAC. The ROM size is exponentially proportional to the desired phase resolution. The huge lookup table not only restricts its maximum operation frequency due to its delays through multiple layers of combinational logic in the decoder, but it also occupies a large area and consumes a large amount of power. The simplest method to reduce the ROM size is to make use of the fact that a sine wave has quarter-wave symmetry. This can be used to cut the ROM size by a factor of four. Although many other ROM-compression methods have been proposed, such as trigonometric approximation and parabolic approximation, the ROM remains the speed bottleneck of the conventional DDS architecture. To reduce the power dissipation and die size, a nonlinear DAC with sine-weighted current cells can be employed, and a ROM-less DDS can be constructed, as shown in Figure 2.12. The nonlinear DAC weights its current cells with sinusoidal data and, hence, converts the digital phase information directly into an analog amplitude waveform, as will be explained more fully in Chapter 10. Thus, the ROM is completely removed, and the performance of the DDS is significantly improved [107–115].

2.7 Direct Analog Frequency Synthesizers

PLL synthesis can achieve high output frequency with low spurious components, yet it suffers from slow switching times [116–121]. An alternate technique is known as direct analog frequency synthesis (DAS), in which a number of different output frequencies are derived from the main reference by mixing, division, and frequency multiplication. Direct analog frequency synthesis can offer excellent spectral purity and high switching speed, but they are usually bulky and expensive and have high power consumption. As a result, this technique is not economical or suitable for portable equipment. Such synthesizers are used only when absolutely necessary in applications such as medical imaging and military radar systems.

Figure 2.13 illustrates a direct analog frequency synthesizer, where a bank of crystal oscillators generates various reference frequencies that are further multiplied to their harmonic frequencies and divided to their subharmonic frequencies.

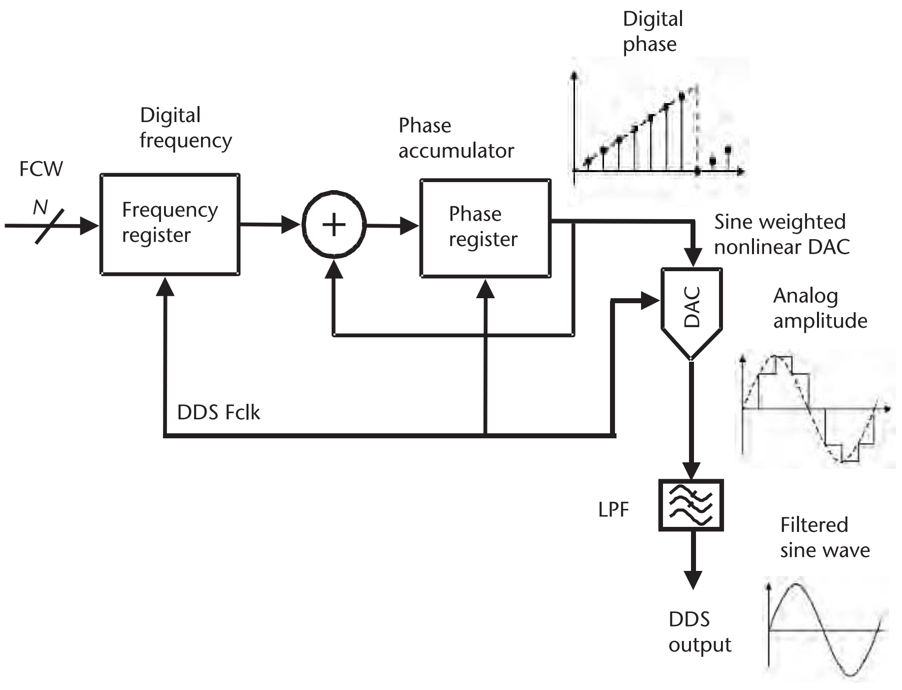


Figure 2.12 ROM-less DDS with a nonlinear, sinusoidal-weighted DAC.

Mixers are employed to mix the different frequencies to achieve the desired output frequency.

The mixer is a device that combines two signals at different frequencies to give an output signal that can be at the sum or difference frequency, as selected by a suitable output filter, which suppresses the other sideband. The process can be extended further, multiplying the number of frequencies by increasing the number of crystals in each oscillator crystal bank. The crystals can be switched rapidly by diodes or by switching the power supply to separate oscillators, one for each crystal. The main problem with the mixer synthesizer is the difficulty of avoiding or filtering out the strong, higher-order mixing products that are produced by the nonlinearities inherent in any RF mixer circuit. Mixer synthesizers were developed in the late 1950s and early 1960s but never achieved widespread use.

2.8 Hybrid Frequency Synthesizers

Various frequency synthesis schemes mentioned above, such as PLL, DDS, and DAS, can be combined to form a hybrid frequency synthesizer to utilize their advantages fully. For instance, a DDS can be embedded into a multiloop hybrid PLL arrangement where, with careful design, the best of both worlds can be achieved. The DDS can give the combination synthesizer small step size, good phase noise, and faster switching, while the PLL can synthesize the carrier frequency at an ultrahigh frequency and can be also used to generate the DDS clock frequency.

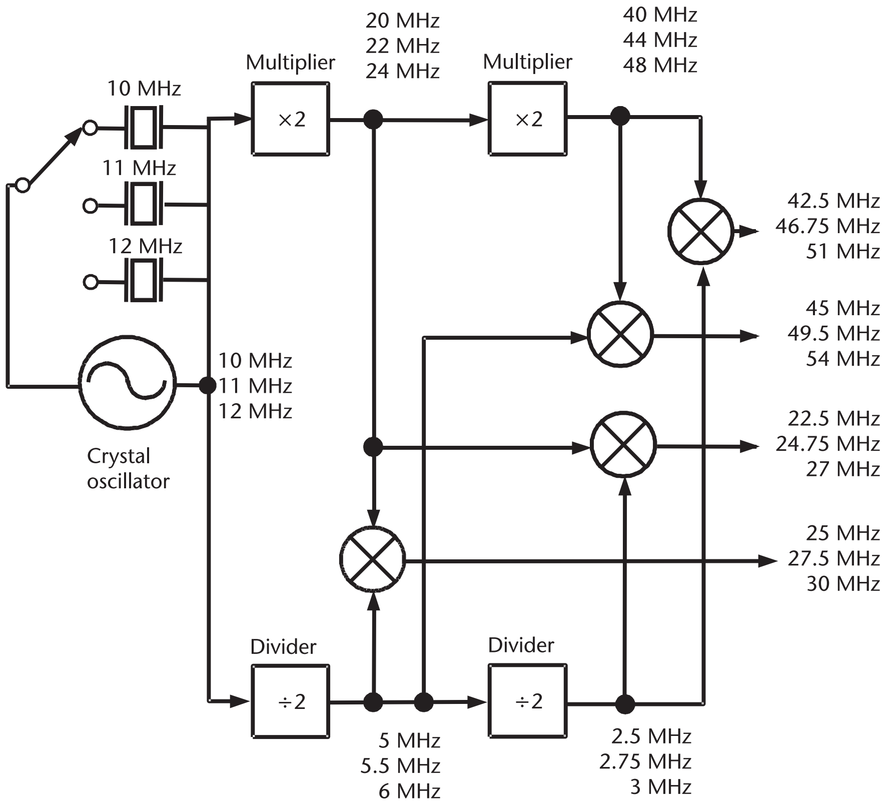


Figure 2.13 A direct analog frequency synthesizer using mixers, multipliers, dividers, and a bank of crystal references.

Figure 2.14 illustrates an example of a hybrid frequency synthesizer with a DDS for fine tune and a PLL for coarse tune. The architecture includes a DDS to generate the fine-tune carrier frequency and to modulate the baseband signal onto this intermediate (IF) carrier frequency. If a ROM-less DDS architecture is employed, the DDS clock frequency can reach 10 GHz and beyond. That allows the system to synthesize and modulate the IF up to 5 GHz. The hybrid synthesizer utilizes a low-noise PLL synthesizer to generate both the DDS clock and the carrier frequency at X-band, K-band, and even Ka-band. The PLL synthesizer includes a quadrature VCO to generate the quadrature carriers without using lossy and narrowband polyphase networks. The DDS modulation waveform configurations can include chirp, step frequency, MSK, PM, AM, QAM, and other hybrid modulations. The modulated IF frequency is mixed with the carrier frequency using quadrature mixers with image rejection.

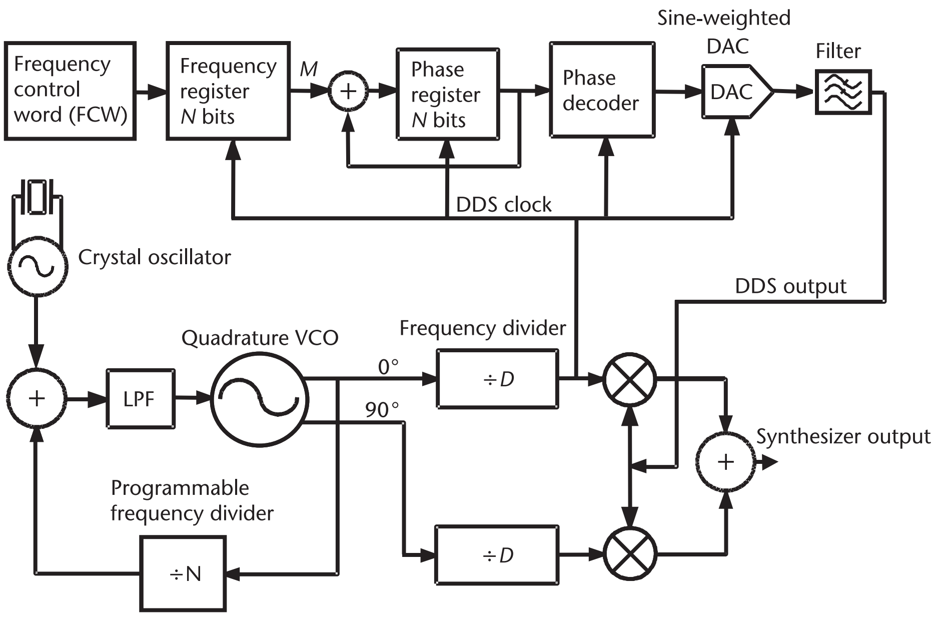


Figure 2.14 A hybrid frequency synthesizer with a DDS for fine tune and a PLL for coarse tune.

References

[1] Larson, L. E., (ed.), *RF and Microwave Circuit Design for Wireless Communications*, Norwood, MA: Artech House, 1997.

[2] Craninckx, J., and M. Steyaert, *Wireless CMOS Frequency Synthesizer Design*, Dordrecht, the Netherlands: Kluwer Academic Publishers, 1998.

[3] Gorsky-Popiel, J., (ed.), *Frequency Synthesis and Applications*, New York: IEEE Press, 1975.

[4] Kroupa, V. F., *Frequency Synthesis*, New York: John Wiley & Sons, 1973.

[5] Kroupa, V. F., "Low-Noise Microwave Synthesizer Design Principles," in *Direct Digital Frequency Synthesizers*, New York: IEEE Press, 1999, pp. 175–180.

[6] Manassewitch, V., *Frequency Synthesizers Theory and Design*, New York: John Wiley & Sons, 1987.

[7] Noordanus, J., "Frequency Synthesizers—A Survey of Techniques," *IEEE Trans. Comm. Tech.*, Vol. 17, No. 2, April 1969, pp. 257–271.

[8] Reinhardt, V. S., et al., "A Short Survey of Frequency Synthesizer Techniques," *Frequency Control Symposium*, Philadelphia, PA, May 1986, pp. 355–365.

[9] Rohde, U. L., *Digital Frequency Synthesizers: Theory and Design*, Upper Saddle River, NJ: Prentice Hall, 1983.

[10] Rohde, U. L., *Microwave and Wireless Synthesizers: Theory and Design*, New York: John Wiley & Sons, 1997.

[11] Slinn, K. R., et al., "Low-Noise Synthesizers for Radar and Communications," *IEE Proceedings*, Vol. 130, H, No.7, December 1983 (Great Britain).

[12] Lindsay, W. C., *Phase-Locked Loops*, New York: IEEE Press, 1986.

[13] Razavi, B., *Design of Integrated Circuits for Optical Communications*, New York: McGraw-Hill, 2002.

[14] Razavi, B., *Monolithic Phase-Locked Loops and Clock Recovery Circuits*, New York: Wiley-IEEE Press, 1996.

[15] Best, R. E., *Phase-Locked Loops: Theory, Design, and Applications*, New York: McGraw-Hill, 1984.

[16] Blanchard, A., *Phase-Locked Loops: Applications to Coherent Receiver Design*, New York: John Wiley & Sons, 1976.

[17] Manassewitz, V., *Frequency Synthesizers: Theory and Design*, 2nd ed., New York: John Wiley & Sons, 1980.

[18] Gardner, F. M., *Phaselock Techniques*, New York: John Wiley & Sons, 1979.

[19] Egan, W. F., *Frequency Synthesis by Phase Lock*, New York: John Wiley & Sons, 2000.

[20] Wolaver, D. H., *Phase-Locked Loop Circuit Design*, Upper Saddle River, NJ: Prentice Hall, 1991.

[21] Crawford, J. A., *Frequency Synthesizer Design Handbook*, Norwood, MA: Artech House, 1994.

[22] Lee, H., et al., "A Σ-Δ Fractional-*N* Frequency Synthesizer Using a Wide-Band Integrated VCO and a Fast AFC Technique for GSM/GPRS/WCDMA Applications," *IEEE J. Solid-State Circuits*, Vol. 39, July 2004, pp. 1164–1169.

[23] Leung, G., and H. Luong, "A 1-V 5.2 GHz CMOS Synthesizer for WLAN Applications," *IEEE J. Solid-State Circuits*, Vol. 39, November 2004, pp. 1873–1882.

[24] Rhee, W., B. Song, and A. Ali, "A 1.1-GHz CMOS Fractional-*N* Frequency Synthesizer with a 3-b Third-Order $\Sigma\Delta$ Modulator," *IEEE J. Solid-State Circuits*, Vol. 35, October 2000, pp. 1453–1460.

[25] Lo, C., and H. Luong, "A 1.5-V 900-MHz Monolithic CMOS Fast-Switching Frequency Synthesizer for Wireless Applications," *IEEE J. Solid-State Circuits*, Vol. 37, April 2002, pp. 459–470.

[26] Ahola, R., and K. Halonen, "A 1.76-GHz 22.6 mW $\Delta\Sigma$ Fractional-*N* Frequency Synthesizer," *IEEE J. Solid-State Circuits*, Vol. 38, January 2003, pp. 138–140.

[27] Heng, C., and B. Song, "A 1.8-GHz CMOS Fractional-*N* Frequency Synthesizer with Randomizer Multiphase VCO," *IEEE J. Solid-State Circuits*, Vol. 38, June 2003, pp. 848–854.

[28] Park, C., O. Kim, and B. Kim, "A 1.8-GHz Self-Calibrated Phase-Locked Loop with Precise I/Q Matching," *IEEE J. Solid-State Circuits*, Vol. 36, May 2001, pp. 777–783.

[29] Klepser, B., M. Scholz, and E. Götz, "A 10-GHz SiGe BiCMOS Phase-Locked-Loop Frequency Synthesizer," *IEEE J. Solid-State Circuits*, Vol. 37, March 2002, pp. 328–335.

[30] Pellerano, S., et al., "A 13.5-mW 5-GHz Frequency Synthesizer with Dynamic-Logic Frequency Divider," *IEEE J. Solid-State Circuits*, Vol. 39, February 2004, pp. 378–383.

[31] Leenaerts, D., et al., "A 15-mW Fully Integrated I/Q Synthesizer for Bluetooth in 0.18-μm CMOS," *IEEE J. Solid-State Circuits*, Vol. 38, July 2003, pp. 1155–1162.

[32] Kan, T., G. Leung, and H. Luong, "A 2-V 1.8-GHz Fully Integrated CMOS Dual-Loop Frequency Synthesizer," *IEEE J. Solid-State Circuits*, Vol. 37, August 2002, pp. 1012–1020.

[33] Aytur, T., and B. Razavi, "A 2-GHz, 6-mW BiCMOS Frequency Synthesizer," *IEEE J. Solid-State Circuits*, Vol. 30, December 1995, pp. 1457–1462.

[34] Chen, W., et al., "A 2-V 2.3/4.6-GHz Dual-Band Frequency Synthesizer in 0.35-μm Digital CMOS Process," *IEEE J. Solid-State Circuits*, Vol. 39, January 2004, pp. 234–237.

[35] Yan, W., and H. Luong, "A 2-V 900-MHz CMOS Dual-Loop Frequency Synthesizer for GSM Receivers," *IEEE J. Solid-State Circuits*, Vol. 36, February 2001, pp. 204–216.

[36] Shu, K., et al., "A 2.4-GHz Monolithic Fractional-*N* Frequency Synthesizer with Robust Phase-Switching Prescaler and Loop Capacitance Multiplier," *IEEE J. Solid-State Circuits*, Vol. 38, June 2003, pp. 866–874.

[37] Shu, Z., K. Lee, and B. Leung, "A 2.4-GHz Ring-Oscillator-Based CMOS Frequency Synthesizer with a Fractional Divider Dual-PLL Architecture," *IEEE J. Solid-State Circuits*, Vol. 39, March 2004, pp. 452–462.

[38] McMahill, D., and C. Sodini, "A 2.5-Mb/s GFSK 5.0-Mb/s 4-FSK Automatically Calibrated Σ–Δ Frequency Synthesizer," *IEEE J. Solid-State Circuits*, Vol. 37, January 2002, pp. 18–26.

[39] Lam, C., and B. Razavi, "A 2.6-GHz/5.2-GHz Frequency Synthesizer in 0.4-μm CMOS Technology," *IEEE J. Solid-State Circuits*, Vol. 35, May 2000, pp. 788–794.

[40] Perrott, M., T. Tewksbury, and C. Sodini, "A 27-mW CMOS Fractional-*N* Synthesizer Using Digital Compensation for 2.5-Mb/s GFSK Modulation," *IEEE J. Solid-State Circuits*, Vol. 32, December 1997, pp. 2048–2060.

[41] Temporiti, E., et al., "A 700-kHz Bandwidth ΣΔ Fractional Synthesizer with Spurs Compensation and Linearization Techniques for WCDMA Applications," *IEEE J. Solid-State Circuits*, Vol. 39, September 2004, pp. 1446–1454.

[42] Dehng, G., et al., "A 900-MHz 1-V CMOS Frequency Synthesizer," *IEEE J. Solid-State Circuits*, Vol. 35, August 2000, pp. 1211–1214.

[43] Lin, T., and W. Kaiser, "A 900-MHz 2.5-mA CMOS Frequency Synthesizer with an Automatic SC Tuning Loop," *IEEE J. Solid-State Circuits*, Vol. 36, March 2001, pp. 424–431.

[44] Rategh, H., H. Samavati, and T. H. Lee, "A CMOS Frequency Synthesizer with an Injection-Locked Frequency Divider for a 5-GHz Wireless LAN Receiver," *IEEE J. Solid-State Circuits*, Vol. 35, May 2000, pp. 780–787.

[45] Hwang, I., S. Song, and S. Kim, "A Digitally Controlled Phase-Locked Loop with a Digital Phase-Frequency Detector for Fast Acquisition," *IEEE J. Solid-State Circuits*, Vol. 36, October 2001, pp. 1574–1581.

[46] Zhang, B., P. Allen, and J. Huard, "A Fast Switching PLL Frequency Synthesizer with an On-Chip Passive Discrete-Time Loop Filter in 0.25-μm CMOS," *IEEE J. Solid-State Circuits*, Vol. 38, October 2003, pp. 855–865.

[47] Da Dalt, N., et al., "A Fully Integrated 2.4-GHz LC-VCO Frequency Synthesizer with 3-ps Jitter in 0.18-μm Standard Digital CMOS Copper Technology," *IEEE J. Solid-State Circuits*, Vol. 37, July 2002, pp. 959–962.

[48] Craninckx, J., and M. S. J. Steyaert, "A Fully Integrated CMOS DCS-1800 Frequency Synthesizer," *IEEE J. Solid-State Circuits*, Vol. 33, December 1998, pp. 2054–2065.

[49] Koo, Y., et al., "A Fully Integrated CMOS Frequency Synthesizer with Charge-Averaging Charge Pump and Dual-Path Loop Filter for PCS- and Cellular-CDMA Wireless Systems," *IEEE J. Solid-State Circuits*, Vol. 37, May 2002, pp. 536–542.

[50] Bax, W. T., and M. A. Copeland, "A GMSK Modulator Using a ΔΣ Frequency Discriminator-Based Synthesizer," *IEEE J. Solid-State Circuits*, Vol. 36, August 2001, pp. 1218–1227.

[51] Bietti, I., et al., "A UMTS ΣΔ Fractional Synthesizer with 200 kHz Bandwidth and –128 dBc/Hz @ 1 MHz Using Spurs Compensation and Linearization Techniques," *Proc. IEEE Custom Integrated Circuits Conference*, San Jose, CA, September 2003, pp. 463–466.

[52] De Muer, B., and M. Steyaert, "A CMOS Monolithic ΔΣ-Controlled Fractional-*N* Frequency Synthesizer for DCS-1800," *IEEE J. Solid-State Circuits*, Vol. 37, July 2002, pp. 835–844.

[53] Riley, T. A., M. Copeland, and T. Kwasniewski, "Delta–Sigma Modulation in Fractional-*N* Frequency Synthesis," *IEEE J. Solid-State Circuits*, Vol. 28, May 1993, pp. 553–559.

[54] Rogers, J. W. M., et al., "A ΔΣ Fractional-*N* Frequency Synthesizer with Multi-Band PMOS VCOs for 2.4 and 5 GHz WLAN Applications," *European Solid-State Circuits Conference (ESSCIRC)*, September 2003, pp. 651–654.

[55] Vaucher, C. S., et al., "A Family of Low-Power Truly Modular Programmable Dividers in Standard 0.35 μm CMOS Technology," *IEEE J. Solid-State Circuits*, Vol. 35, July 2000, pp. 1039–1045.

[56] Kim, B., T. C. Weigandt, and P. R. Gray, "PLL/DLL System Noise Analysis for Low Jitter Clock Synthesizer Design," *Proc. Int. Symp. Circuits and Systems*, London, U.K., 1994, pp. 31–34.

[57] Lee, T. H., et al., "A 2.5V CMOS Delay-Locked Loop for 18 Mbit, 500 Megabyte/s DRAM," *IEEE J. Solid-State Circuits*, Vol. 29, December 1994, pp. 1491–1496.

[58] Jeon, Y., et al., "A 66–333-MHz 12-mW Register-Controlled DLL with a Single Delay Line and Adaptive-Duty-Cycle Clock Dividers for Production DDR SDRAMs," *IEEE J. Solid-State Circuits*, Vol. 39, November 2004, pp. 2087–2092.

[59] Hamamoto, T., et al. "A 667-Mb/s Operating Digital DLL Architecture for 512-Mb DDR SDRAM," *IEEE J. Solid-State Circuits*, Vol. 39, January 2004, pp. 194–206.

[60] Chien, G., and P. R. Gray, "A 900-MHz Local Oscillator Using a DLL-Based Frequency Multiplier Technique for PCS Applications," *IEEE J. Solid-State Circuits*, Vol. 35, December 2000, pp. 1996–1999.

[61] Maillard, X., F. Devisch, and M. Kuijk, "A 900-Mb/s CMOS Data Recovery DLL Using Half-Frequency Clock," *IEEE J. Solid-State Circuits*, Vol. 37, June 2002, pp. 711–715.

[62] Tierney, J., C. M. Rader, and B. Gold, "A Digital Frequency Synthesizer," *IEEE Trans. Audio Electroacoust.*, Vol. AU-19, 1971, pp. 48–57.

[63] Bjerede, B., and G. Fisher, "A New Phase Accumulator Approach to Frequency Synthesis," *Proceedings of the National Aerospace and Electronics Conference*, May 1976, pp. 928–932.

[64] Cole, C., "Design of a Direct Digital Synthesizer," master's thesis, Massachusetts Institute of Technology, Boston, 1982.

[65] Essenwanger, K. A., "Spurious Suppression in Direct Digital Frequency Synthesis by Combined Dithered Accumulator and Sine Approximation Techniques," master's thesis, California State Polytechnic University, Pomona, May 1987.

[66] Essenwanger, K. A., and V. S. Reinhardt, "Sine Output DDSs: A Survey of the State of the Art," *52nd Frequency Control Symposium*, Pasadena, CA, May 1998, pp. 370–378.

[67] Flanagan, M. J., and G. A. Zimmerman, "Spur-Reduced Digital Sinusoid Synthesis," *IEEE Trans. on Comm.*, Vol. 43, No. 7, July 1995, pp. 2254–2262.

[68] Garvey, J. F., and D. Babitch, "An Exact Spectral Analysis of a Number Controlled Oscillator Based Synthesizer," *44th Frequency Control Symposium*, Baltimore, MD, May 1990, pp. 511–521.

[69] Gillette, G. C., "The Digiphase Synthesizer," *Proc. 23rd Frequency Control Symposium*, Atlantic City, NJ, May 1969, pp. 201–210.

[70] Goldberg, B. G., *Digital Frequency Synthesis Demystified: DDS and Fractional-N PLLs*, Eagle Rock, VA: Llh Technology Publishing, 1999.

[71] Grayver, E., and B. Daneshrad, "Direct Digital Frequency Synthesis Using a Modified Cordic," *Proc. of the International Circuits and Systems Conference*, Vol. 5, 1998, pp. 241–245.

[72] Hassun, R., "The Common Denominators in Fractional *N*," *Microwaves & RF*, Vol. 23, June 1984, pp. 107–110.

[73] Hutchinson, B. H., Jr., "Contemporary Frequency Synthesis Techniques," Chapter 1 in J. Gorsky-Popiel, (ed.), *Frequency Synthesis and Applications*, New York: IEEE Press, 1975.

[74] Kroupa, V. F., "Spectral Purity of Direct Digital Frequency Synthesizers," *44th Frequency Control Symposium*, Baltimore, MD, May 1990, pp. 498–510.

[75] Kroupa, V. F., (ed.), *Digital Frequency Synthesis*, New York: IEEE Press, 1999.

[76] Lundgren, R. E., V. S. Reinhardt, and K. W. Martin, "Designs and Architectures for EW/Communications Direct Digital Synthesizers," Research and Development Technical Report, SLCET-TR-0424-F, U.S. Army Laboratory Command, 1st Interim Report, November 1986; Final Report, August 1987.

[77] Nicholas, H. T., III, H. Samueli, and B. Kim, "The Optimization of Direct Digital Frequency Synthesizer Performance in the Presence of Finite Word Length Effects," *42nd Frequency Control Symposium*, Baltimore, MD, June 1988, pp. 357–363.

[78] Nicholas, H. T., III, and H. Samueli, "An Analysis of the Output of Direct Digital Frequency Synthesizers in the Presence of Phase-Accumulator Truncation," *41st Frequency Control Symposium*, Philadelphia, PA, May 1987, pp. 495–502.

[79] Nossen, E. J., "Digital Frequency Synthesis," U.S. Patent 4,206,425, June 3, 1980.

[80] O'Leary, P., and F. Maloberti, "A Direct-Digital Synthesizer with Improved Spectral Performance," *IEEE Trans. on Comm.*, Vol. 39, July 1991, pp. 1045–1048.

[81] Reinhardt, V. S., "Direct Digital Synthesizers," *Proceedings of the 17th PTTI Planning Meeting*, Washington, DC, December 1985.

[82] Reinhardt, V. S., and I. Shahriary, "Spurless Fractional Divider Direct Digital Frequency Synthesizer and Method," U.S. Patent 4,815,018, March 21, 1989.

[83] Reinhardt, V. S., "Method and Apparatus for Reduced Aliasing in Signal Processing," U.S. Patent 4,890,249, December 26, 1989.

[84] Reinhardt, V. S., K. V. Gould, and K. M. McNab, "Randomized Digital/Analog Converter Direct Digital Synthesizer," U.S. Patent 5,014,231, May 7, 1991.

[85] Reinhardt, V. S., "Spur Reduction Techniques in Direct Digital Synthesizers," *Proc. Frequency Control Symposium*, June 1993, pp. 230–241.

[86] Rohde, U., *Digital PLL Frequency Synthesizers—Theory and Design*, New York: Prentice Hall, 1983.

[87] Sunderland, D., et al., "CMOS/SOS Frequency Synthesizer LSI Circuit for Spread Spectrum Communications," *IEEE J. Solid-State Circuits*, Vol. 19, No. 4, August 1984, pp. 497–506.

[88] Tierney, J., et al., "A Digital Frequency Synthesizer," *IEEE Trans. Audio Electroacoust.*, Vol. AU-19, March 1971, p. 48.

[89] Vankka, J., "Spur Reduction Techniques in Sine Output Direct Digital Synthesis," *IEEE International Frequency Control Symposium*, 1996, pp. 951–959.

[90] Vankka, J., et al., "A Direct Digital Synthesizer with an On-Chip D/A-Converter," *IEEE J. Solid-State Circuits*, Vol. 33, February 1998, pp. 218–227.

[91] Volder, J. E., "The CORDIC Trigonometric Computing Techniques," *IRE Trans. on Electronic Computers*, Vol. EC-8, 1959.

[92] Wheatley, C. E., III, and D. E. Phillips, "Spurious Suppression in Direct Digital Synthesizers," in *Direct Digital Frequency Synthesizer*, pp. 119–126, New York: IEEE Press, 1999.

[93] Wheatley, C. E., III, "Digital Frequency Synthesizer with Random Jittering for Reducing Discrete Spectral Spurs," U.S. Patent 4,410,954, October 18, 1983.

[94] Nicholas, H. T., and H. Samueli, "An Analysis of the Output Spectrum of Direct Digital Frequency Synthesizers in the Presence of Phase-Accumulator Truncation," *Proc. 41st Annual Frequency Control Symp.*, Philadelphia, PA, May 1987, pp. 495–502.

[95] O'Leary, P., and F. Maloberti, "A Direct-Digital Synthesizer with Improved Spectral Performance," *IEEE Trans. Comm.*, Vol. 39, No. 7, July 1991, pp. 1046–1048.

[96] Kim, B., H. T. Nicholas, and H. Samueli, "The Optimization of Direct Digital Frequency Synthesizer in the Presence of Finite Word Length Effects," *Proc. 42nd Annual Frequency Control Symp.*, Baltimore, MD, 1988, pp. 357–363.

[97] Madisetti, A., A. Kwentus, and A. Willson, "A 100-MHz, 16-b, Direct Digital Frequency Synthesizer with a 100-dBc Spurious-Free Dynamic Range," *IEEE J. Solid-State Circuits*, Vol. 34, August 1999, pp. 1034–1043.

[98] Song, Y., and B. Kim, "A 14-b Direct Digital Frequency Synthesizer with Sigma-Delta Noise Shaping," *IEEE J. Solid-State Circuits*, Vol. 39, May 2004, pp. 847–851.

[99] Nicholas, H., and H. Samueli, "A 150-MHz Direct Digital Frequency Synthesizer in 1.25-μm CMOS with –90-dBc Spurious Performance," *IEEE J. Solid-State Circuits*, Vol. 26, December 1991, pp. 1959–1969.

[100] Yamagishi, A., et al., "A 2-V, 2-GHz Low-Power Direct Digital Frequency Synthesizer Chip-Set for Wireless Communication," *IEEE J. Solid-State Circuits*, Vol. 33, February 1998, pp. 210–217.

[101] Torosyan, A., D. Fu, and A. Willson, "A 300-MHz Quadrature Direct Digital Synthesizer/ Mixer in 0.25-μm CMOS," *IEEE J. Solid-State Circuits*, Vol. 38, June 2003, pp. 875–887.

[102] Saul, P., and M. Mudd, "A Direct Digital Synthesizer with 100-MHz Output Capability," *IEEE J. Solid-State Circuits*, Vol. 23, June 1988, pp. 819–821.

[103] Vankka, J., et al., "A Direct Digital Synthesizer with an On-Chip D/A-Converter," *IEEE J. Solid-State Circuits*, Vol. 33, February 1998, pp. 218–227.

[104] Nakagawa, T., and H. Nosaka, "A Direct Digital Synthesizer with Interpolation Circuits," *IEEE J. Solid-State Circuits*, Vol. 32, February 1997, pp. 766–770.

[105] Nosaka, H., et al., "A Low-Power Direct Digital Synthesizer Using a Self-Adjusting Phase-Interpolation Technique," *IEEE J. Solid-State Circuits*, Vol. 32, August 2001, pp. 1281–1285.

[106] Jiang, J., and E. Lee, "A Low-Power Segmented Nonlinear DAC-Based Direct Digital Frequency Synthesizer," *IEEE J. Solid-State Circuits*, Vol. 37, October 2002, pp. 1326–1330.

[107] Sodagar, A. M., and G. R. Lahiji, "Mapping ROM Phase to Sine-Amplitude in Direct Digital Frequency Synthesizers Using Parabolic Approximation," *IEEE Trans. Circuits Syst. II*, Vol. 47, December 2000, pp. 1452–1457.

[108] Mortezapour, S., and E. K. F. Lee, "Design of Low-Power Frequency Synthesizer Using Nonlinear Digital-to-Analog Converter," *IEEE J. Solid-State Circuits*, Vol. 34, October 1999, pp. 1350–1359.

[109] Nakamura, Y., et al., "A 10-b 70-MS/s CMOS D/A Converter," *IEEE J. Solid-State Circuits*, Vol. 26, April 1991, pp. 637–642.

[110] Vorenkamp, P., et al., "A 1 Gs/s, 10 bit Digital-to-Analog Converter," *ISSCC Dig. Tech Papers*, San Francisco, CA, February 1994, pp. 52–53.

[111] Yamashina, M., and H. Yamada, "MOS Current Mode Logic MCML Circuit for Low-Power GHz Processors," *NEC Res. Develop.*, Vol. 36, No. 1, January 1995, pp. 54–63.

[112] Vandenbussche, J., et al., "A 14 Bit 100 MSamples Update Rate 42 Random Walk CMOS D/A Converter," *ISSCC Dig. Tech Papers*, San Francisco, CA, February 1999, pp. 146–147.

[113] Van den Bosch, A., M. Steyaert, and W. Sansen, "SFDR-Bandwidth Limitations for High Speed High Resolution Current Steering CMOS D/A Converters," *IEEE International Conference on Electronics, Circuits and Systems*, Pafos, Cyprus, September 1999, pp. 1193–1196.

[114] Gutierrez-Aitken, A., et al., "Ultrahigh-Speed Direct Digital Synthesizer Using InP DHBT Technology," *IEEE J. Solid State Circuits*, September 2002, pp. 1115–1121.

[115] Dai, F. F., et al., "A Low Power 5 GHz Direct Digital Synthesizer Implemented in SiGe Technology," *IEEE 5th Topical Meeting on Silicon Monolithic Integrated Circuits in RF Systems*, Atlanta, GA, September 2004, pp. 21–24.

[116] Driscoll, M. M., "Phase Noise Performance of Analog Frequency Dividers," *Proc. Frequency Control Symposium*, Denver, CO, June 1989, pp. 342–348.

[117] Egan, W. E., "Modeling Phase Noise in Frequency Dividers," *IEEE Trans. on Ultrasonics, Ferroelectrics, and Frequency Control*, Vol. 37, July 1990, pp. 307–315.

[118] Harrison, R. G., "Theory of Regenerative Frequency Dividers Using Double-Balanced Mixers," *IEEE MTT-S Digest*, Vol. 1, June 1989, pp. 267–270.

[119] Llopis, O., et al., "Phase Noise Performance of Microwave Analog Frequency Dividers-Applications to the Characterization of Oscillators up to the MM-Wave Range," *Frequency Control Symposium*, Pasadena, CA, May 1998, pp. 550–554.

[120] Wachnick, R. A., T. J. Bucelot, and G. P. Li, "Degradation of Bipolar Transistors under High Current Stress at 300K," *J. Appl. Phys.*, Vol. 63, No. 9, 1988, pp. 4734–4740.

[121] Rubiola, E., M. Olivier, and J. Groslambert, "Phase Noise in the Regenerative Frequency Dividers," *IEEE Trans. I&M*, Vol. 41, June 1992, pp. 353–360.

CHAPTER 3

System-Level Overview of PLL-Based Frequency Synthesis

3.1 Introduction

There are many ways to realize synthesizers; possibly the most common is based on a PLL [1–7]. PLL-based synthesizers can be further subdivided by which type of a division is used in the feedback path. The division ratio *N* can be either an integer or a fractional number. If the number is fractional, then the synthesizer is called a fractional-*N* synthesizer. This type of synthesizer can be further distinguished by the method used to control the divide ratio, for example by a $\Sigma\Delta$ controller or by some other technique. In this chapter, analysis is done with a general *N* without giving the details of how *N* is implemented; thus, the analysis is applicable both to integer-*N* and fractional-*N* synthesizers. An example of a synthesizer not based on a PLL is the direct-digital synthesizer.

3.2 PLLs (Example of a Feedback System)

Figure 3.1 shows a block diagram for a PLL. In brief, the PLL is a feedback system that forces the divided-down VCO output phase to follow the reference signal phase. That is, it is a negative feedback loop with phases as the input and output signals. The loop is composed of a phase detector, a lowpass filter, a VCO, and a divider. The phase detector, which is the summing block of the feedback system, is used to compare output phase θ_o to reference phase θ_R. The LPF controls the loop's dynamic behavior, which is usually a linear transfer function that is placed in the system to control the settling time, transient response, and so forth. The VCO generates the output signal, and the divider divides the VCO output signal back down to the same frequency as the input. Since f_o is divided down from the VCO output, it follows that f_{VCO} is $N \cdot f_{\text{ref}}$. We use a feedback system based on phase rather than frequency because, in any feedback loop without infinite dc gain, there is always an error (finite error signal) between the input (reference) and the output. Thus, if we used a frequency-locked loop, then, in most cases, there would be an error in the output frequency, and it would not track the input as precisely as does a loop based on phase. The input reference in wireless communications is a quartz crystal. These crystals are low in cost and can be made to resonate with extreme accuracy at a particular frequency determined by the physical properties

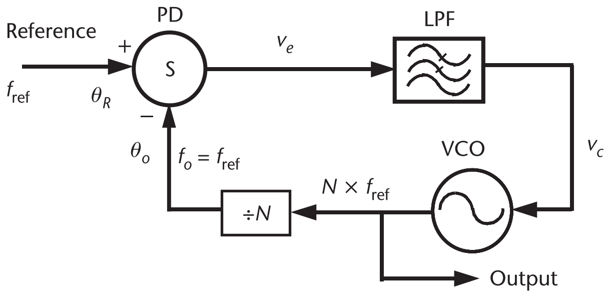


Figure 3.1 A basic block diagram of a frequency synthesizer.

of the crystal. Unfortunately, they can only resonate at frequencies as high as about 100 MHz, and, therefore, cannot be used directly as an LO in RF applications. The other disadvantage to using a crystal directly is that there is no convenient way to tune its frequency. This is one of the main reasons that frequency synthesizers have become so popular. If the divider block is implemented using circuitry such that the divide ratio is programmable, then a range of frequencies can be obtained without the need to change the reference frequency.

3.3 PLL Components

We will now briefly look at the basic components needed to make a PLL-based synthesizer and their governing equations. This chapter will provide only a very basic introduction to these components and consider only the most common forms of these circuits. Later chapters will deal with each of these circuits and variations on them in much more detail. For now, enough information will be given to allow the system-level analysis of a PLL.

3.3.1 VCOs and Dividers

At the most basic level, all VCOs will have an output frequency with some dependence on the control voltage (or sometimes control current) as shown in Figure 3.2. Note that the curve is not always linear (actually, it is hardly ever linear), but, for the moment, we will assume that it is. Also, note that the control voltage can usually be adjusted between ground and the power-supply voltage and that, over that range, the VCO frequency will move between some minimum and some maximum value.

Here, V_c is the nominal control voltage coming from the loop filter, and ω_{nom} is the nominal frequency at this nominal voltage. Usually, when considering loop dynamics, we only consider frequency deviations away from the nominal frequency ω_{VCO} and voltage deviations away from the nominal voltage v_c. Thus, we can write the oscillating frequency as

$$\omega_o = \omega_{\text{nom}} + \omega_{\text{VCO}} = \omega_{\text{nom}} + K_{\text{VCO}} v_c \tag{3.1}$$

where

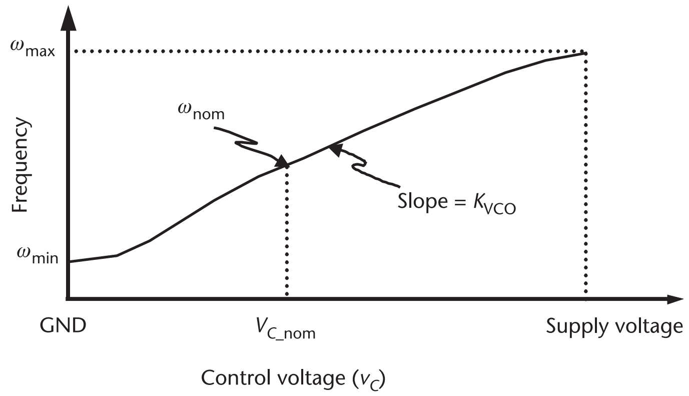


Figure 3.2 A typical VCO characteristic.

$$v_c = v_C - V_{\text{C\_nom}} \tag{3.2}$$

In addition, if we remove the "dc" part of the equation and only consider the part that is changing, we are left with

$$\omega_{\text{VCO}} = K_{\text{VCO}} v_c \tag{3.3}$$

However, we would like to have an expression relating input voltage to output phase since the output of the VCO ultimately goes to the phase detector. To relate frequency ω to phase θ, we note that

$$\omega = \frac{d\theta}{dt} \tag{3.4}$$

Therefore, the output phase of the VCO can be given as

$$\theta_{\text{VCO}} = \int \omega_{\text{VCO}}\, dt = K_{\text{VCO}} \int_0^t v_c(\tau)\, d\tau \tag{3.5}$$

In the Laplace domain, this becomes

$$\frac{\theta_{\text{VCO}}(s)}{v_c(s)} = \frac{K_{\text{VCO}}}{s} \tag{3.6}$$

Thus, we have the desired equation for the transfer function of the VCO block. Note that, for the purposes of system behavior, the divider can be thought of as an extension of the VCO. The output phase after the divider is simply

$$\frac{\theta_o}{v_c} = \frac{1}{N} \cdot \frac{K_{\text{VCO}}}{s} \tag{3.7}$$

Example 3.1: Example of VCO Excess Phase

Assume two VCOs are identical. One has a constant $V_{C\_nom} = 1.5\text{V}$ bias applied to its control line, and the other has a 1.5-V bias applied to its control line, but this voltage experiences a transient step of 300 mV for a period of 100 ps, 200 ps after the VCOs have been started. Each VCO has a K_{VCO} of 100 MHz/V and an f_{nom} of 2 GHz. Determine the total accumulated phase of each VCO after a period of 1 ns. What is the excess phase of the VCO that experiences the transient?

Solution: Each VCO will operate nominally at a frequency of 2 GHz, which corresponds to a period of 500 ps. Thus, in 1 ns, the first VCO will accumulate a total phase of 4π rad. The second VCO will run 300 mV · 100 MHz/V = 30 MHz faster for a time of 100 ps. Thus, during this time, $f_{\text{VCO}} = 30$ MHz. As a result, it accumulates an excess phase of

$$\text{Excess phase} = (2\pi \cdot 30 \text{ MHz}) \cdot 100 \text{ ps} = 1.88 \cdot 10^{-2} \text{ rad}$$

or 1.1°, in addition to the 4π of phase that it accumulated due to the dc voltage. Thus, it has been disturbed from its nominal operating point by 1.1°. This excess phase is the only phase that we generally consider when doing small-signal analysis for PLLs.

3.3.2 Phase Detectors

A phase detector produces an output signal proportional to the phase difference of the signals applied to its inputs. The inputs and outputs can be sine waves, square waves, or other periodic signals, not necessarily having a 50% duty cycle. The output signal could be a current or voltage, and it could have multiple frequency components. Since the dc value is the component of interest, the phase detector is typically followed by some sort of filter. Thus, the equation that describes a phase detector is

$$v_e(s) = K_{\text{phase}}[\theta_R(s) - \theta_o(s)] \tag{3.8}$$

provided that the phase detector is a continuous-time circuit (which is often not the case in IC implementations). The output of the phase detector, $v_e(s)$, is often also called the error voltage and is seen to be proportional to the difference of the input phases with proportionality constant K_{phase}. This is a linearized equation, often valid only over limited range. Another comment that can be made about phase detectors is that, often, they respond in some way to a frequency difference as well. In such a case, the circuit is often referred to as a phase-frequency detector (PFD).

The phase detector can be as simple as an "exclusive or" (XOR) gate or an "exclusive nor" (XNOR) gate. Typical phase detectors are usually more complicated; for example, flip-flops form the basis of tristate phase detectors. Phase detectors are also often combined with charge pumps.

3.3.2.1 The Exclusive NOR Gate as a Phase Detector

Figure 3.3 shows the XNOR gate used as a phase detector and the dc output voltage versus phase difference. It can be seen that the output is maximum when

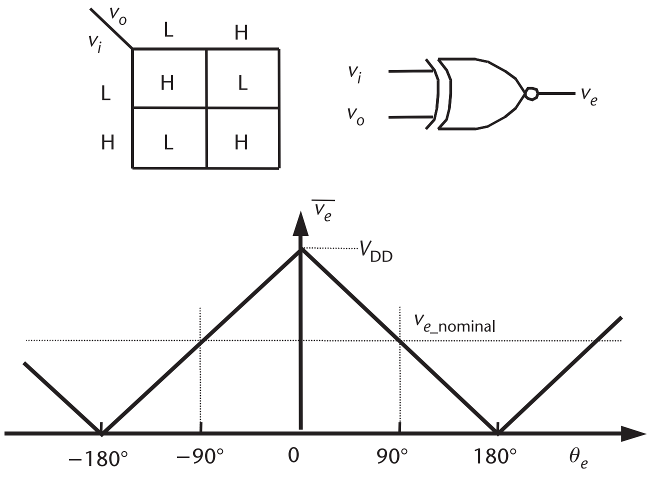


Figure 3.3 The XNOR phase detector and its average output voltage versus phase input.

the inputs are in phase and minimum when the inputs are out of phase. The midpoint output occurs for inputs at 90°; thus, this is the nominal input phase. That is, for minimum and maximum output voltages equally below and above ground, respectively, the output will be 0V when the inputs are 90° apart.

Figure 3.4 shows the time domain waveforms of the XNOR gate at phase angles of 90°, 45°, and 135°, which help to explain how the graph in Figure 3.3 was obtained. As expected, at 90° the output has a 50% duty-cycle waveform with an average voltage halfway between the two voltage extremes. An average value for any other phase difference can be determined in a similar manner.

Once the graph in Figure 3.3 has been constructed, it is easy to see that, for this phase detector, K_{phase} is given by

$$K_{\text{phase}} = \pm \frac{V_{\text{DD}}}{\pi} \tag{3.9}$$

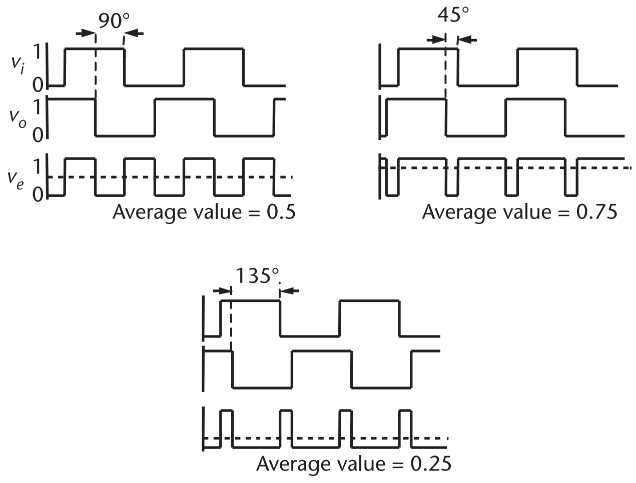


Figure 3.4 Waveforms of the XNOR phase detector.

Note that, depending on the value of the phase difference, the slope can be positive or negative. This very useful property can ensure that accidents with feedback polarity in PLLs do not cause instability.

3.3.2.2 PFD and Charge Pump

A much more common type of phase detector is the tristate phase detector, often called a phase frequency detector (PFD), which has two outputs, as shown in Figure 3.5. If the reference phase (v_R) is ahead of the output phase (v_o), then the circuit produces an UP signal that tells the VCO to speed up and, therefore, advance its phase to catch up with the reference phase. Conversely, if the reference phase is lagging behind the output phase, it produces a DN signal that tells the VCO to slow down and, therefore, retard its phase to match the reference phase. If the reference and output are in phase, then the phase detector does not produce an output. The UP and DN signals are also sometimes called v_U and v_D, respectively.

The two digital signals produced by a PFD have to be converted back into an analog control signal at the input of the VCO, and the circuit most commonly used to do this is called a charge pump. A charge pump is made of two controllable current sources connected to a common output, also shown in Figure 3.5. The outputs from the phase detector turn on one of the two currents, which either charge or discharge capacitors attached to the VCO input.

The PFD circuitry will be discussed in Chapter 6, but here is a quick description of its operation based on the state diagram shown in Figure 3.6. Transitions happen

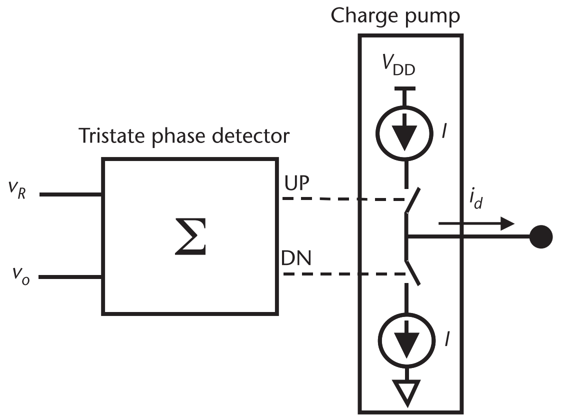


Figure 3.5 Tristate phase detector and charge pump.

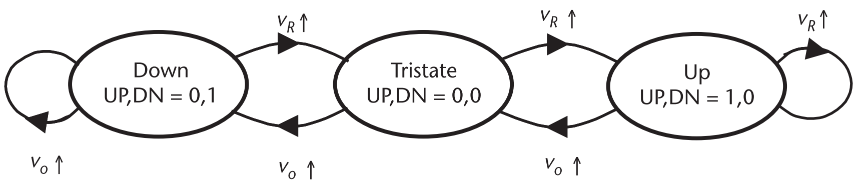


Figure 3.6 PFD state diagram.

only on the rising edge of v_o or v_R. Let us assume that we start in the middle state, the tristate where both outputs are zero. Then, depending on which edge arrives first, the PFD moves either to the up or down state. If the reference edge arrives first, the output needs to catch up, so the up switch turns on to charge up the output. It stays up until the other edge comes along; thus, the average output current from the charge pump depends on how far apart the two signals are. On the other hand, if the reference is lagging behind the output, then the output is too fast and needs to be slowed down. This causes a down pulse to be generated, and, as a result, current flows out of the charge pump, discharging the output. Current flows for the length of time τ between the output edge and the reference edge. If the period is T, the average output current is

$$i_d = I\frac{\tau}{T} = \left(\frac{I}{2\pi}\right)(\theta_R - \theta_o) \tag{3.10}$$

Thus, K_{phase} for this phase detector is

$$K_{\text{phase}} = \frac{I}{2\pi} \tag{3.11}$$

where I is the current that flows through the controllable current sources in the charge pump when they are on.

The operation of the PFD and current sources is shown in Figure 3.7. The movement from state to state is controlled by the rising edge only, so the pulse width of the input signals is not important. We have shown the pulses as being narrow, but it would work equally well using wider pulses. In the diagram, it is shown that for the same phase difference between v_o and v_R, the output current depends on where the operation starts. In Figure 3.7(a), the v_o edge comes first, resulting in down pulses and a negative average-output current. In Figure 3.7(b), the v_R edge comes first, resulting in up pulses and a positive average-output current. We note also that if the v_o pulse were delayed (moved towards the right) in Figure 3.7(a), the down pulses would become narrower, resulting in average current closer to zero. In Figure 3.7(b), for the same delay of the v_o pulses, the up pulses become wider, and the average current moves closer to I. Note that the short pulses

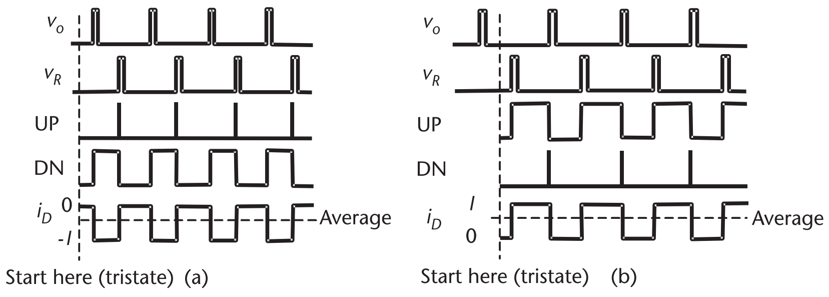


Figure 3.7 Operation of PFD and current sources, with starting points (a) before v_o and (b) before v_R.

associated with UP in Figure 3.7(a) and DN in Figure 3.7(b) are realistic. They are the result of details of the PFD design and will be discussed in Chapter 6.

If this average output current is now plotted as a function of the phase difference, with v_R taken as the reference, the result can be interpreted as the transfer function of the phase detector and charge pump and is shown in Figure 3.8. We note that any positive phase (for example, 60°, for which the output current would be $I \times 60/360$) could be interpreted as the equivalent negative phase (for example, +60° is equivalent to −300°, for which the current is equal to $-I \times 300/360$). Thus, for every phase difference, there are two possible interpretations, and this is shown by the solid and dashed lines in Figure 3.8. We note that this is equivalent to starting the phase detector in a different state or at a different time, as was shown in Figure 3.7.

To illustrate how this phase detector can also be used as a frequency detector, Figure 3.9 shows waveforms for two different input frequencies. We have assumed that we start in tristate. Since the output pulse v_o occurs first, down pulses DN occur, which would result in negative output current pulses i_d and an average negative output current, shown by the dotted line. However, since the frequencies are different, the pulse width is changing, in this case becoming narrower and the average current is moving towards zero. Eventually, the phase detector experiences a second reference pulse v_R before the output pulse v_o and moves into the up state, and up current pulses i_d result. From then on, the phase detector output states will be either tristate or in the up state, so only positive current is ever provided. In this way, it can be seen that, for a reference frequency higher than the output frequency, average current is positive. Similarly, for a reference frequency lower

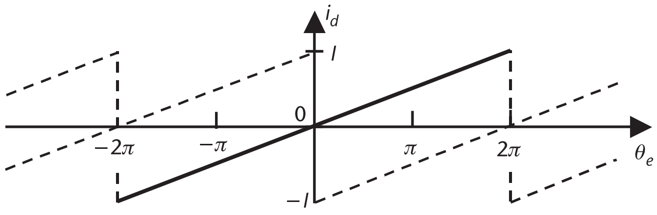


Figure 3.8 Average output current versus phase for PFD and charge pump.

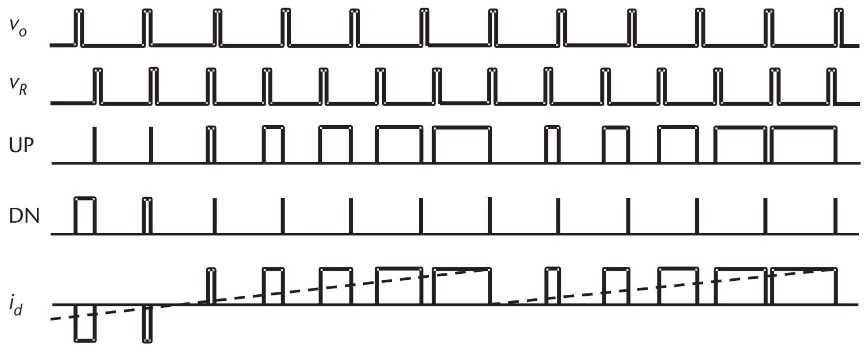


Figure 3.9 Output pulses for inputs at different frequencies.

than the output frequency, the average output current is always negative (except, of course, possibly for a short time at startup). Thus, with the correct feedback polarity, this current can be used to correct the VCO frequency until it is the same as the reference frequency, and the loop is locked.

3.3.3 The Loop Filter

Normally, VCOs are controlled by voltage and not current. Thus, generally, we need a method to turn the current produced by the charge pump back into a voltage. In addition, lowpass filtering is needed since it is not desirable to feed pulses into the VCO. This is usually done by dumping the charge produced by the charge pump onto the terminals of a capacitor. As we will show later, a simple capacitor all by itself does not yield a stable loop, so a combination of capacitors and resistors is used. This part of the PLL is typically called the loop filter. One of the most common loop filters used is shown in Figure 3.10. Note that PLLs that do not use charge pumps have loop filters as well. In addition to turning the current back into the voltage, loop filters are also the components most commonly used to control system-level loop dynamics.

The frequency response of the PFD, charge pump, and loop filter is mainly determined by the loop filter. The frequency response of the network (seen in Figure 3.10) will now be analyzed, starting with the admittance of the capacitor and resistor circuit.

$$Y = sC_2 + \frac{1}{R + \frac{1}{sC_1}} = sC_2 + \frac{sC_1}{sC_1R + 1} = \frac{sC_2(sC_1R + 1) + sC_1}{sC_1R + 1} \tag{3.12}$$

This admittance can be used to determine v_c the control voltage of the VCO as follows:

$$v_c = \frac{i_d}{Y} = \frac{K_{\text{phase}}(\theta_R - \theta_o)(sC_1R + 1)}{sC_2(sC_1R + 1) + sC_1} = \frac{K_{\text{phase}}(\theta_R - \theta_o)(1 + sC_1R)}{s(C_1 + C_2)(1 + sC_sR)} \tag{3.13}$$

where

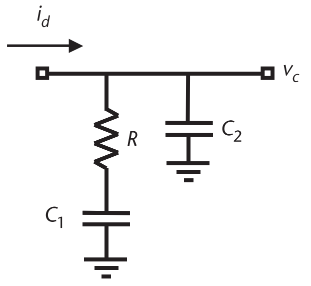


Figure 3.10 A typical loop filter.

$$C_s = \frac{C_1 C_2}{C_1 + C_2} \quad \text{and} \quad K_{\text{phase}} = \frac{I}{2\pi}$$

Figure 3.11 shows the frequency response of the charge pump and filter as given in (3.13) We note that at low frequencies, the response is dominated by the zero in the transfer function; thus, the circuit acts like an integrator. Also note that this has been derived in continuous time and, as long as the pulses are much faster than any changes of interest at the output, this is a reasonable assumption.

3.4 Continuous-Time Analysis for PLL Synthesizers

The *s* domain model for a synthesizer is shown in its most general form in Figure 3.12. Here, any loop filter (in combination with a charge pump) is simply shown

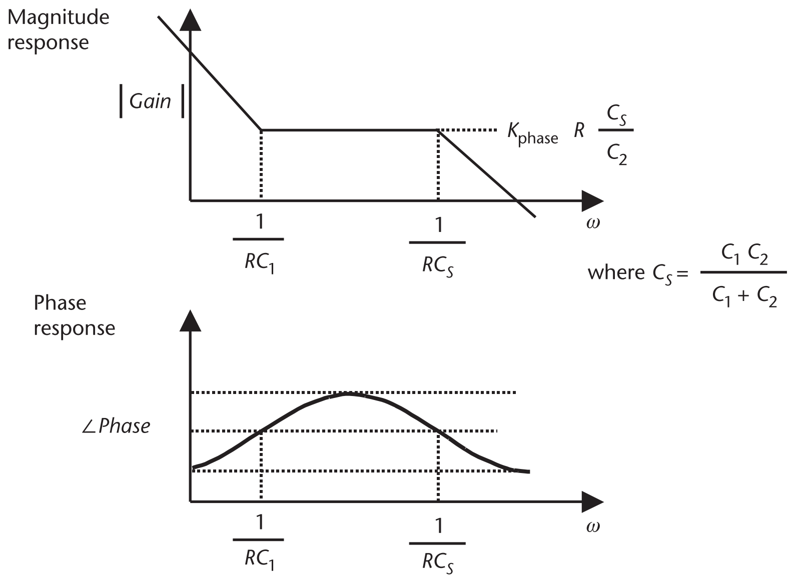


Figure 3.11 PFD, charge pump, and loop filter frequency response.

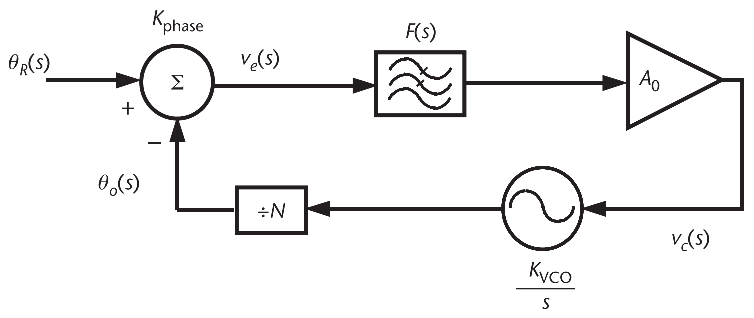


Figure 3.12 Complete loop in the frequency domain.

as $F(s)$, and the dc gain is brought out explicitly as term A_o. We can therefore derive the basic loop transfer function for the loop.

3.4.1 Simplified Loop Equations

The overall transfer function is

$$\frac{\theta_o}{\theta_R} = \frac{\dfrac{A_o K_{\text{phase}} F(s)}{N} \cdot \dfrac{K_{\text{VCO}}}{s}}{1 + \dfrac{A_o K_{\text{phase}} F(s)}{N} \cdot \dfrac{K_{\text{VCO}}}{s}} = \frac{KF(s)}{s + KF(s)} \tag{3.14}$$

where K is given by

$$K = \frac{A_o K_{\text{phase}} K_{\text{VCO}}}{N} \tag{3.15}$$

Now (3.14) is the most general PLL loop equation, and, for specific loops, it differs only in the form that $F(s)$ takes. For instance, in a first-order loop, $F(s)$ is simply equal to one. In this case, the loop equation becomes

$$\frac{\theta_o}{\theta_R} = \frac{K}{s + K} \tag{3.16}$$

Note that for this first-order loop, for a phase ramp (change in frequency), the phase error is not zero because there are not enough integrators in the loop. Since zero phase error is often highly desired, and due to its lack of flexibility, this loop is not often used in practice.

A much more common PLL is the second-order PLL, an example of which is the PFD/CP-based PLL. A typical second-order PLL has a loop filter with a transfer function of

$$F(s) = \frac{\tau s + 1}{s} \tag{3.17}$$

A PFD/CP-based PLL with the loop filter, as previously discussed, is an example of a second-order PLL, as will now be shown. Figure 3.13 shows the most common system-level configuration. In this case (assuming for the moment that we ignore C_2), the impedance of the loop filter is

$$F(s) = R + \frac{1}{sC_1} = \frac{sC_1 R + 1}{sC_1} \tag{3.18}$$

Note that this transfer function is an impedance since this stage converts current to voltage, which is not a typical transfer function but works here. Thus, we can substitute this back into (3.14) and, therefore, find

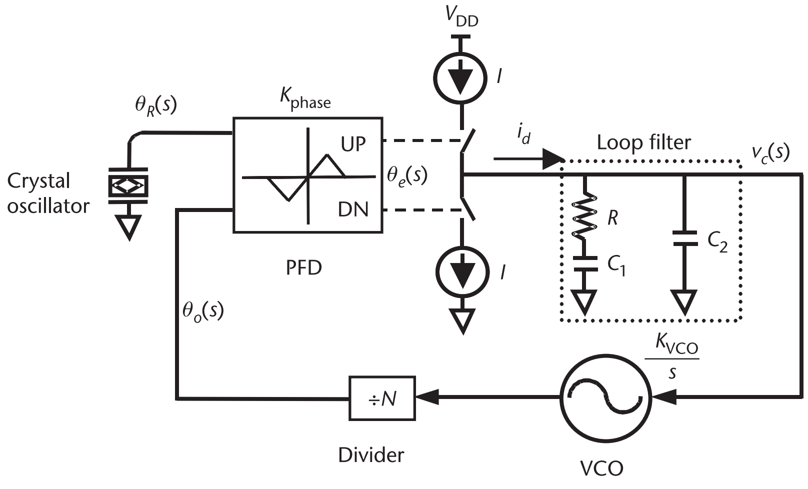


Figure 3.13 A frequency synthesizer implemented with a charge pump and PFD.

$$\frac{\theta_o}{\theta_R} = \frac{\frac{IK_{VCO}}{2\pi \cdot N}\left(R + \frac{1}{sC_1}\right)}{s + \frac{IK_{VCO}}{2\pi \cdot N}\left(R + \frac{1}{sC_1}\right)} = \frac{\frac{IK_{VCO}}{2\pi \cdot NC_1}(RC_1 s + 1)}{s^2 + \frac{IK_{VCO}}{2\pi \cdot N}Rs + \frac{IK_{VCO}}{2\pi \cdot NC_1}} \tag{3.19}$$

Thus, for this PLL, we get a second-order transfer function with a zero. Note that the purpose of R can be seen directly from this equation. If R is set equal to zero, it can be seen by inspection of (3.19) that the poles of this equation will sit on the $j\omega$ axis, and the loop will oscillate or be on the verge of oscillating. From (3.19), expressions for the loop dynamics can be determined. The natural frequency of the loop is given by

$$\omega_n = \sqrt{\frac{IK_{VCO}}{2\pi \cdot NC_1}} \tag{3.20}$$

The damping constant is given by

$$\zeta = \frac{R}{2}\sqrt{\frac{IK_{VCO}C_1}{2\pi \cdot N}} \tag{3.21}$$

Often the resistor and capacitor values are to be determined for a known damping constant and natural frequency. It is straightforward to solve these two equations for these variables:

$$C_1 = \frac{IK_{VCO}}{2\pi \cdot N\omega_n^2} \tag{3.22}$$

and

$$R = 2\zeta\sqrt{\frac{2\pi \cdot N}{IK_{\text{VCO}}C_1}} = \zeta\frac{4\pi \cdot N\omega_n}{IK_{\text{VCO}}} \tag{3.23}$$

From the above, it can be shown that (3.19) can be rewritten in a general form as

$$\frac{\theta_o}{\theta_R} = \frac{\omega_n^2\left(\dfrac{2\zeta}{\omega_n}s + 1\right)}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{3.24}$$

This demonstrates that there is a relationship between the pole and zero locations.

Note that it is easy to determine the transfer function, even if the output is taken from other places in the loop. For instance, it is often interesting to look at the control voltage going into the VCO. In this case, the system transfer function becomes

$$\frac{v_C}{\theta_R} = \frac{\dfrac{I \cdot s}{2\pi \cdot C_1}(RC_1 s + 1)}{s^2 + \dfrac{IK_{\text{VCO}}}{2\pi \cdot N}Rs + \dfrac{IK_{\text{VCO}}}{2\pi \cdot NC_1}} = \frac{\dfrac{N\omega_n^2}{K_{\text{VCO}}}s\left(\dfrac{2\zeta}{\omega_n}s + 1\right)}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{3.25}$$

This expression contains an extra *s* in the numerator. This makes sense because the control voltage is proportional to the frequency of the VCO, which is the derivative of the phase of the output. We can also write an expression for the output frequency (as given by the control voltage) as a function of the input frequency, noting that frequency is the derivative of phase and starting from (3.25):

$$\frac{v_C}{\theta_R s} = \left[\frac{\dfrac{I \cdot s}{2\pi \cdot C_1}(RC_1 s + 1)}{s^2 + \dfrac{IK_{\text{VCO}}}{2\pi \cdot N}Rs + \dfrac{IK_{\text{VCO}}}{2\pi \cdot NC_1}}\right]\frac{1}{s} = \left[\frac{\dfrac{N\omega_n^2}{K_{\text{VCO}}}s\left(\dfrac{2\zeta}{\omega_n}s + 1\right)}{s^2 + 2\zeta\omega_n s + \omega_n^2}\right]\frac{1}{s} \tag{3.26}$$

$$\frac{v_C}{\omega_R} = \frac{\dfrac{N\omega_n^2}{K_{\text{VCO}}}\left(\dfrac{2\zeta}{\omega_n}s + 1\right)}{s^2 + 2\zeta\omega_n s + \omega_n^2}$$

which is nearly identical to the expression for phase transfer function θ_o/θ_R given by (3.24).

3.4.2 PLL System Frequency Response and Bandwidth

Figure 3.14 is a plot of the closed-loop transfer function for the PFD/CP-based PLL, which is described by (3.19) and (3.24) for different values of the damping constant. This diagram shows that the loop's 3-dB bandwidth is highly dependent

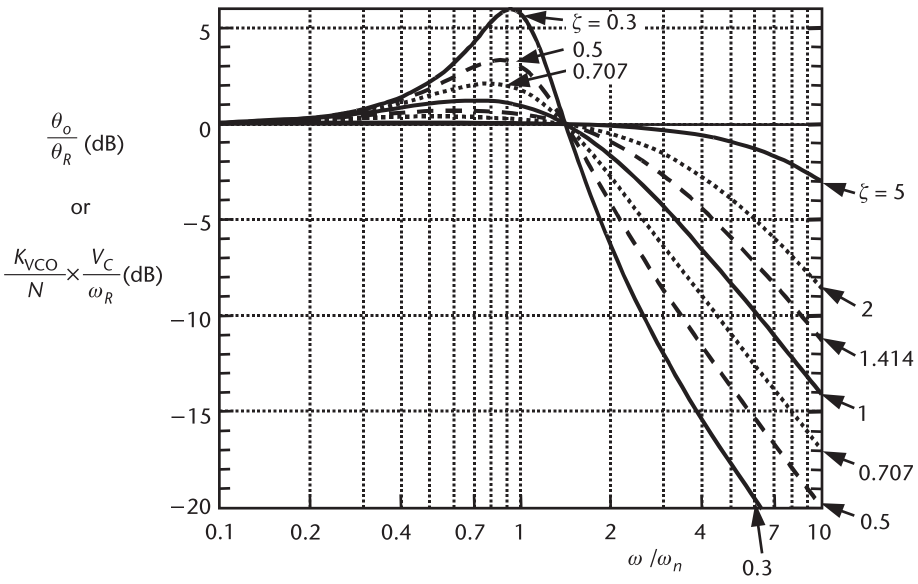


Figure 3.14 PLL frequency response of the closed-loop transfer function of a high-gain, second-order loop. Note, the graph is for either of the two functions shown.

on the damping constant. Using a method similar to that presented in Section A.4.1, it can be shown that the 3-dB bandwidth of this system is given by

$$\omega_{3\,\text{dB}} = \omega_n \sqrt{1 + 2\zeta^2 + \sqrt{4\zeta^4 + 4\zeta^2 + 2}} \tag{3.27}$$

Since this equation can be tedious without a calculator, two equations sometimes used to approximate this are

$$\begin{aligned} \omega_{3\,\text{dB}} &\approx 2\zeta\omega_n && \zeta > 1.5 \text{ (Approximation \#1)} \\ \omega_{3\,\text{dB}} &\approx \left(1 + \zeta\sqrt{2}\right)\omega_n && \zeta < 1.5 \text{ (Approximation \#2)} \end{aligned} \tag{3.28}$$

The validity of these two approximations can also be illustrated with a simple plot, and they are compared in Figure 3.15.

3.4.3 Complete Loop Transfer Function, Including C_2

Note that if C_2 is included, this adds a high-frequency pole to the system. Normally, this capacitor is chosen to be about one-tenth of the value of C_1 and is included to clean up high-frequency ripple on the control line. If this capacitor is included, then the following expression for open-loop gain can be derived:

$$\left(\frac{\theta_o}{\theta_R}\right)_{\text{open loop}} = \frac{K_{VCO} K_{\text{phase}} (1 + sC_1R)}{s^2 N (C_1 + C_2)(1 + sC_sR)} \tag{3.29}$$

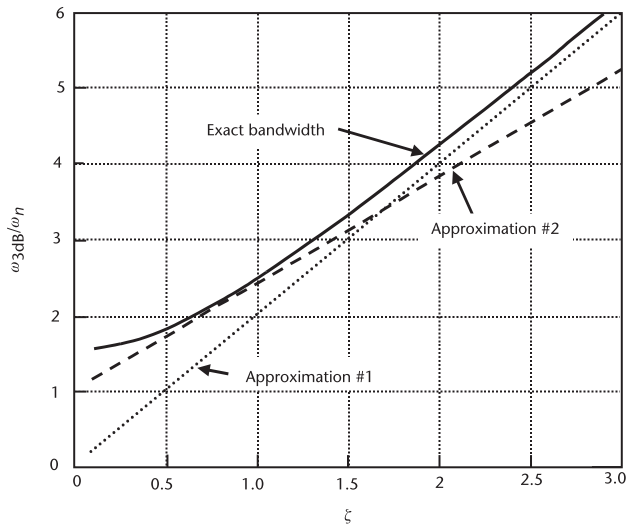


Figure 3.15 Comparison of bandwidth formulas for PLLs.

Here, C_s is the series combination of C_1 and C_2. This is plotted in Figure 3.16, which shows a low-frequency slope of –40 dB/dec and 180° of phase shift. After the zero, the slope is –20 dB/dec, and the phase heads back towards 90° of phase shift. After the high-frequency pole, the slope is again –40 dB/dec, and the phase approaches 180°. Note that the dashed lines in the graph show the response of the system if the capacitor C_2 is not included. For optimal stability (maximum phase margin in the system), the unity gain point should be at the geometric mean of the zero and the high-frequency pole since this is the location where the phase

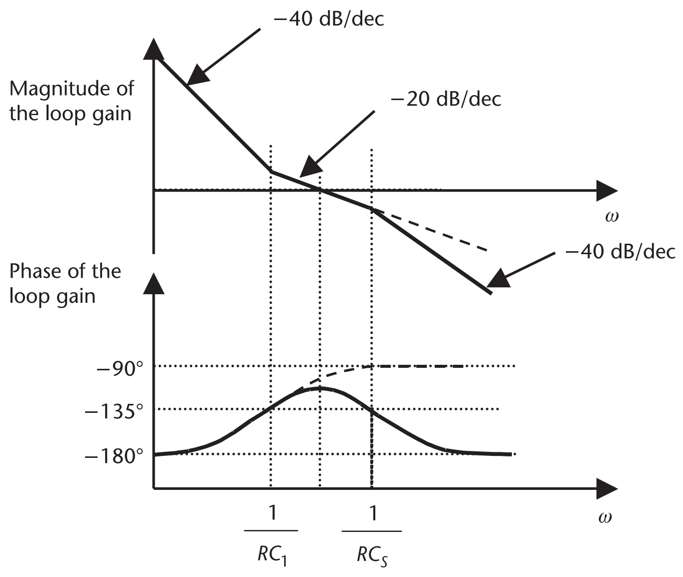


Figure 3.16 Open-loop magnitude and phase response. Note that the dotted line shows response if the high-frequency pole is not included.

shift is furthest from 180°. Some may wonder, after seeing this plot, if the system is actually unstable at dc because, at this point, the phase shift is 180°, and the gain is greater than one. In fact, the system is stable. A full stability analysis, like plotting the closed-loop poles, would show this.

The closed-loop gain with C_2 is given by

$$\frac{\theta_o}{\theta_R} = \frac{K_{\mathrm{VCO}} K_{\mathrm{phase}} (1 + sC_1 R)}{s^2 N(C_1 + C_2)(1 + sC_s R) + K_{\mathrm{VCO}} K_{\mathrm{phase}} (1 + sC_1 R)} \tag{3.30}$$

Thus, one can now estimate all of the parameters of the loop. Figure 3.16 shows that if the zero and the high-frequency pole are relatively far apart, then, up to the unity gain point, the loop parameters are nearly the same whether or not the high-frequency pole is included. There is, however, a slight decrease of phase margin (in the diagram, from about 75° to about 65°).

A cautionary note about the choice of C_2 should also be sounded. It may not always be appropriate to use a value of one-tenth C_1 for the value of C_2. It has been assumed in this section that, most of the time, C_2 does not significantly change the loop dynamics; however, R increases at high ζ, and, in this case, the impedance of the series combination of C_1 and R may become comparable to the impendence of C_2 at the loop natural frequency. If this happens, then the equations derived in this section will become increasingly inaccurate, in which case, there will be no choice but to reduce the value of C_2 to less than $C_1/10$.

3.5 Discrete-Time Analysis for PLL Synthesizers

The preceding linear, continuous-time analysis of the synthesizer is not valid under all conditions. If the loop bandwidth is increased so that it becomes a significant fraction of the reference frequency, the previous analyses become increasingly inaccurate. For this reason, it is sometimes necessary to treat the synthesizer system as the discrete-time control system that it truly is [5]. To do this, we must consider the PFD, which in this case is the sampling element. We assume that, in lock, the PFD produces only narrow impulses at the reference frequency. Note that because the charge pump behaves like an integrator, it has infinite gain at dc. Thus, as long as the frequency deviation is small, this high gain drives the phase error towards zero, and the output pulses will be narrow. Therefore, it acts like an ideal sampler. The loop filter holds the charge dumped onto it in each cycle, so, in this system, it acts as a hold function. Hence, this is a sampled system, and the s domain combination of the VCO, divider, PFD, and loop filter must be converted into their z domain equivalents (be careful to remember to multiply the functions together before converting them into the z domain, as discussed in Appendix A). Thus, the open-loop transfer function, including the sampling action of these four blocks, is (ignoring C_2) (see Figure 3.17)

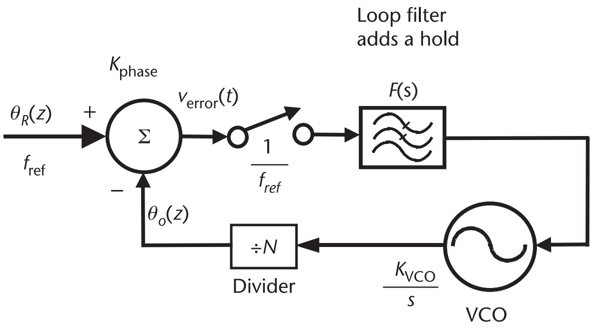


Figure 3.17 Discrete-time, system-level diagram for a synthesizer.

$$
\begin{aligned}
G_{\text{OL}}(s) &= F(s) \cdot K_{\text{phase}} \cdot \frac{K_{\text{VCO}}}{N \cdot s} \cdot \left(\frac{1 - e^{-sT}}{s}\right) \\
&= \left(R + \frac{1}{sC_1}\right) \cdot \frac{K_{\text{VCO}} K_{\text{phase}}}{N \cdot s} \cdot \left(\frac{1 - e^{-sT}}{s}\right) \qquad (3.31) \\
&= \omega_n^2 \left(\frac{\frac{2\zeta}{\omega_n} s + 1}{s^2}\right) \cdot \left(\frac{1 - e^{-sT}}{s}\right)
\end{aligned}
$$

where T is the period of the reference.

Now $G_{\text{OL}}(s)$ is converted to $G_{\text{OL}}(z)$ using Table A.1 in Appendix A:

$$
G_{\text{OL}}(z) = \frac{\omega_n^2 T^2}{2}\left(1 + \frac{4\zeta}{\omega_n T}\right) \cdot \left[\frac{z - \frac{4\zeta - \omega_n T}{4\zeta + \omega_n T}}{(z-1)^2}\right] = K\left[\frac{z - \alpha}{(z-1)^2}\right] \qquad (3.32)
$$

where the open-loop zero α, which has a value between −1 and +1, depending on the reference period, is given by

$$
\alpha = \frac{4\zeta - \omega_n T}{4\zeta + \omega_n T} \qquad (3.33)
$$

and the open-loop gain, which depends on the reference period, is given by

$$
K = \frac{\omega_n^2 T^2}{2}\left(1 + \frac{4\zeta}{\omega_n T}\right) \qquad (3.34)
$$

Now the closed-loop gain of the system can also be determined as

$$
G(z) = \frac{K(z - \alpha)}{z^2 + (K - 2)z + (1 - \alpha K)} \qquad (3.35)
$$

Starting from either the open-loop gain and using root locus, or else directly using the closed-loop transfer function, the pole locations as a function of the reference period T can be sketched and are shown in Figure 3.18. Note that depending on the specific parameters, this plot will change slightly, but the basic shape of the root locus will remain the same. The point of greatest interest here is that point at which the reference period is increased to some critical value and the system becomes unstable, as one of the poles moves outside the unit circle. At, or even close to, this period, the s domain analysis discussed in the previous section will be increasingly inaccurate. Note that the reference frequency will not normally be this low in the design of a PLL, but the designer must be aware of this so that the assumptions of the previous section are not violated.

Now the poles of (3.35) are given by

$$\text{Poles} = 1 - \frac{K}{2} \pm \frac{1}{2}\sqrt{(K-2)^2 - 4(1-\alpha K)} \tag{3.36}$$

The pole that has the larger positive value is not of concern because it will never leave the unit circle. However, it is of interest to find out when

$$1 - \frac{K}{2} - \frac{1}{2}\sqrt{(K-2)^2 - 4(1-\alpha K)} = -1 \tag{3.37}$$

Skipping a number of steps (primarily because they are boring), this will happen when

$$K(1+\alpha) = 4 \tag{3.38}$$

Taking this expression and substituting back in for K and α, the critical period for which the loop will go unstable, T_{US}, is

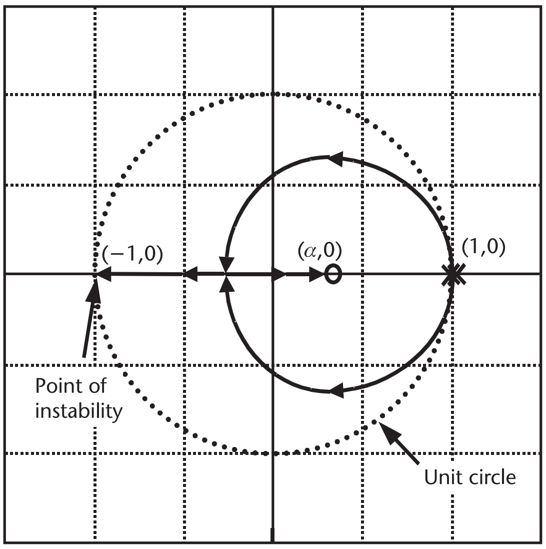


Figure 3.18 Sketch of closed-loop pole location as a function of increasing reference period.

$$T_{\text{US}} = \frac{1}{\omega_n \zeta} \tag{3.39}$$

Noting that

$$T_{\text{US}} = \frac{2\pi}{\omega_{\text{ref\_crt}}} \tag{3.40}$$

where $\omega_{\text{ref\_crt}}$, the reference frequency at which the loop goes unstable, can be determined to be

$$\omega_{\text{ref\_crt}} = 2\pi\zeta\omega_n \tag{3.41}$$

Therefore,

$$\frac{\omega_{\text{ref}}}{\omega_n} \geq 2\pi\zeta \tag{3.42}$$

So, for instance, in the case of $\zeta = 0.707$, this ratio must be greater than 4.4. Therefore, for a reference frequency of 40 MHz, if the loop natural frequency is set any higher than 9.1 MHz, the loop will go unstable. A ratio often quoted as being "safe" is 10:1 [6].

3.6 Transient Behavior of PLLs

The two previous sections derived linear s domain equations that describe the PLL as a classic feedback system and more complicated z domain equations. However, the behavior of a real PLL is much more complex than either of these two analyses can explain. This is because, until the loop starts tracking the phase of the input, or, alternatively, if there is a very large step or ramp in the phase of the input, the loop's output phase may not be able to follow the input phase. This is primarily due to the limitations of the phase detector, which has a narrow linear range. For example, the tristate PFD has a linear range of $\pm 2\pi$. If an event at the input occurs that causes the phase error to exceed 2π, then the loop will experience a nonlinear event: cycle slipping. Remember that in the previous analysis, it was assumed that the phase detector was linear. This nonlinear event will cause a transient response that cannot be predicted by the theory of the previous section. The loop will, of course, work to correct the phase and force the VCO to track the input once more. When the loop goes into this process, it is said to be in acquisition mode as it is trying to acquire phase lock but has not done so yet. Note that acquisition also happens when the PLL is first powered since the VCO and reference will be at a random phase and will probably have a frequency difference. In extreme cases, the VCO may even be forced beyond its linear range of operation, which may result in the loop losing lock indefinitely. These situations will now be explored. First, the case in which the loop is in lock and experiences no cycle slipping will be considered.

3.6.1 Linear Transient Behavior

Here, the linear transient response of the most common PLL presented in this chapter will be considered further. This section discusses only the *s* domain response, which, under most normal operating conditions, is sufficient. However, the *z* domain equivalent of this analysis could be undertaken with the aid of the discussion in Appendix A. For linear transient behavior, the phase error, rather than the output phase, is needed, so a different transfer function has to be derived for the system shown in Figure 3.13. The result is

$$\frac{\theta_e}{\theta_R} = \frac{s^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{3.43}$$

In this section, we will see the response to an input frequency step $\Delta\omega$. Since the input is described by phase, we take the phase equivalent of a frequency step, which is equivalent to a ramp of phase (we note that phase is the integral of frequency, and the integral of a step is a ramp). Thus, the input is described by

$$\theta_R = \frac{\Delta\omega}{s^2} \tag{3.44}$$

This input in (3.44), when multiplied by the transfer function (3.43), results in

$$\theta_e = \frac{\Delta\omega}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{3.45}$$

Then, the inverse Laplace transform is taken, with the following results:

$$\theta_e(t) = \frac{\Delta\omega}{\omega_n}\left[\frac{\sinh \omega_n\sqrt{\zeta^2 - 1}\,t}{\sqrt{\zeta^2 - 1}}\right]e^{-\zeta\omega_n t} \quad \zeta > 1 \tag{3.46}$$

$$\theta_e(t) = \frac{\Delta\omega}{\omega_n}\,\omega_n t \cdot e^{-\omega_n t} \quad \zeta = 1 \tag{3.47}$$

$$\theta_e(t) = \frac{\Delta\omega}{\omega_n}\left[\frac{\sin \omega_n\sqrt{1 - \zeta^2}\,t}{\sqrt{1 - \zeta^2}}\right]e^{-\zeta\omega_n t} \quad \zeta < 1 \tag{3.48}$$

These results are plotted in Figure 3.19 for various values of the damping constant.

It can be seen that a damping constant of 0.707 to 1 results in the fastest settling (reduction of phase error to zero). Depending on the required level of settling, one can determine the settling time. To the accuracy of the above diagram, settling is better than 99% complete when $\omega_n t = 7$ for $\zeta = 0.707$. Thus,

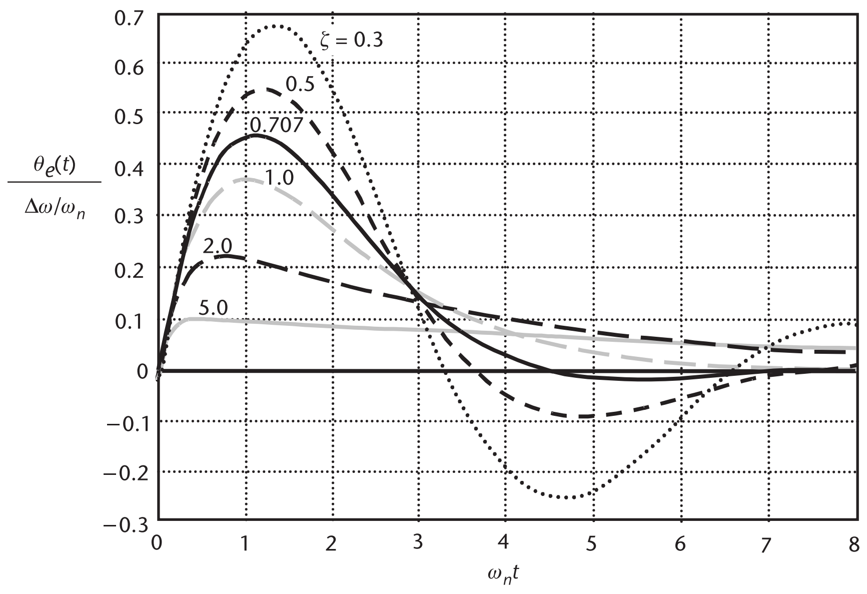


Figure 3.19 Error for frequency-step, high-gain, second-order loop. Note, $\theta_\epsilon = \theta_R - \theta_o$.

given a required settling time, one can calculate the required natural frequency. To prevent the reference frequency from feeding through to the VCO, the loop bandwidth, as shown in Figure 3.14 and estimated in (3.28), must be significantly less than the reference frequency. In fact, the extra capacitor in the loop filter has been added in order to provide attenuation at the reference frequency.

It is also interesting to look at the control voltage:

$$\frac{V_C}{\omega_R} = \frac{\frac{N\omega_n^2}{K_{VCO}}\left(\frac{2\zeta}{\omega_n}s + 1\right)}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{3.49}$$

In this case, again, we apply a step in frequency so that

$$\omega_R = \frac{\Delta\omega}{s} \tag{3.50}$$

Note that this equation is given in the frequency, rather than the phase domain, so s is raised to unity power in the denominator. Therefore, the control voltage is given by

$$V_C = \frac{\frac{N\omega_n^2}{K_{VCO}}\left(\frac{2\zeta}{\omega_n}s + 1\right)}{s^2 + 2\zeta\omega_n s + \omega_n^2} \cdot \frac{\Delta\omega}{s} = \frac{\frac{2\zeta \cdot N\omega_n}{K_{VCO}} \cdot \Delta\omega}{s^2 + 2\zeta\omega_n s + \omega_n^2} + \frac{\frac{N\omega_n^2}{K_{VCO}} \cdot \frac{\Delta\omega}{s}}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{3.51}$$

Now the first term is simply a scaled version of the previous expression, and the second term is the integral of the first term. Therefore, the transient expression for the control voltage is given by

$$V_C(t) = \frac{\zeta \cdot N\Delta\omega}{K_{VCO}} \left[\frac{\sinh \omega_n \sqrt{\zeta^2 - 1}\, t}{\sqrt{\zeta^2 - 1}} \right] e^{-\zeta\omega_n t} - \frac{N\Delta\omega}{K_{VCO}} \left[\cosh \omega_n \sqrt{\zeta^2 - 1}\, t \right] e^{-\zeta\omega_n t} + \frac{N\Delta\omega}{K_{VCO}} \quad \zeta > 1 \tag{3.52}$$

$$V_C(t) = \frac{N\Delta\omega}{K_{VCO}} \omega_n t \cdot e^{-\omega_n t} - \frac{N\Delta\omega}{K_{VCO}} \cdot e^{-\omega_n t} + \frac{N\Delta\omega}{K_{VCO}} \quad \zeta = 1 \tag{3.53}$$

$$V_C(t) = \frac{\zeta \cdot N\Delta\omega}{K_{VCO}} \left[\frac{\sin \omega_n \sqrt{1 - \zeta^2}\, t}{\sqrt{1 - \zeta^2}} \right] e^{-\zeta\omega_n t} - \frac{N\Delta\omega}{K_{VCO}} \left[\cos \omega_n \sqrt{1 - \zeta^2}\, t \right] e^{-\zeta\omega_n t} + \frac{N\Delta\omega}{K_{VCO}} \quad \zeta < 1 \tag{3.54}$$

These expressions are plotted in Figure 3.20. Interestingly, from this expression, it looks like high ζ is best for fast settling; however, it should be noted that, even though the frequency appears to lock quickly, there is still a long period before the system is phase locked. Therefore, although these plots may be useful for comparison with the control voltage (which is more readily available from simulation), they can also be misleading.

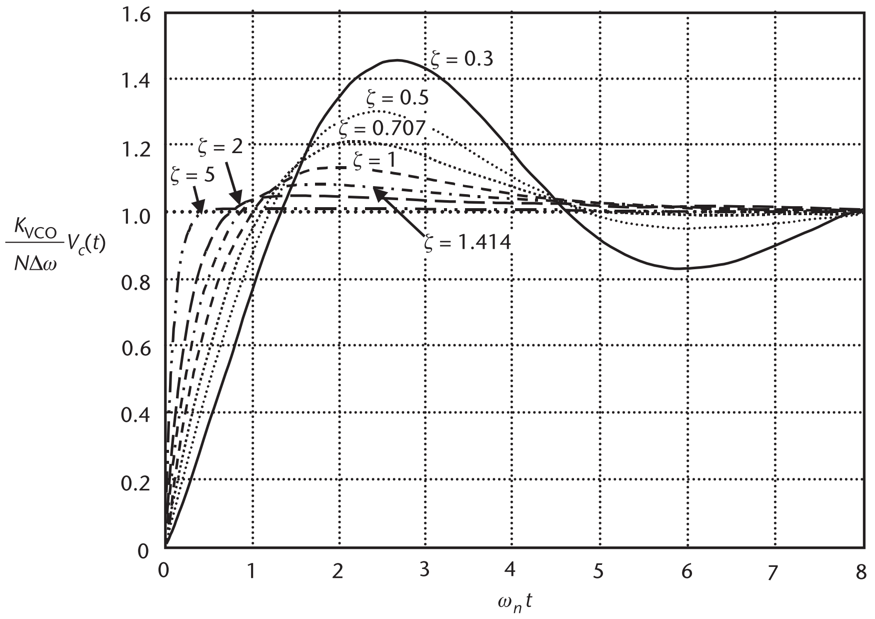


Figure 3.20 Control voltages for frequency-step, high-gain, second-order loop.

Example 3.2: Limits of the Theory So Far

Assume that a synthesizer is designed with a charge pump, PFD, and loop filter like the ones considered thus far in this chapter. Assume that the loop filter is designed so that the system has a damping constant of 0.707, and a 3-dB bandwidth of 150 kHz. What is the maximum frequency step at the input such that the theory so far is still able to predict the behavior of the system? Provided this condition is met, how long will it take the system to settle from a frequency step?

Solution: This is a classic question that the authors have been using to torment undergraduate students for years now. Besides being good exam fodder, it also illustrates a very important point (and all you students just thought we were sadistic, right?).

First, we compute the natural frequency of the loop using approximation #2 from (3.28):

$$\omega_n \approx \frac{\omega_{3\text{ dB}}}{\left(1+\zeta\sqrt{2}\right)} = \frac{2\pi \cdot 150\text{ kHz}}{2} = 2\pi \cdot 75\text{ kHz (Approximation \#2)}$$

Now, referring to Figure 3.19, the maximum normalized phase error to a frequency step is about 0.46 for $\zeta = 0.707$. Therefore, the maximum phase error is

$$\theta_{e\_\max} = 0.46\frac{\Delta\omega}{\omega_n}$$

The maximum phase error that the PFD can withstand is 2π. Therefore, the largest frequency step that the system can handle is

$$\Delta\omega_{\max} = \frac{\theta_{e\_\max}\omega_n}{0.46} = \frac{2\pi(2\pi \cdot 75\text{ kHz})}{0.46} = 6.43\frac{\text{Mrad}}{s} = 1.02\text{ MHz}$$

If the frequency step is any larger than this, then the PLL will lose lock and cycle slip, and the transient response will no longer look like Figure 3.19. If it is smaller than this, Figure 3.19 should do a fair job of predicting the result. In this case, it will take a normalized time of $\omega_n t = 7$ before the transient settles, or about 14.9 μs.

Example 3.3: Integer-N Synthesizer Designed for Settling Time

Design an integer-*N* synthesizer to operate at 2.4 GHz to 2.4835 GHz and which must be able to settle from a frequency step in 225 μs (ignoring cycle slipping, which will be discussed in the next section). The channel spacing for the radio is 1 MHz. These are similar specifications to those of a Bluetooth radio. After the system is designed, determine how much the loop filter attenuates the reference signal.

Solution: For 1-MHz channels, the reference frequency is 1 MHz. Using a damping constant of 0.707, then the settling time is $\omega_n t = 7$ from Figure 3.19. Then, substituting $t = 225\ \mu$s results in $\omega_n = 7/225\ \mu\text{s} = 31.11$ krad/s, or 4.95 kHz. This results in a bandwidth of about 10 kHz from Figure 3.14. Now, loop gain K, divider values (N), and component limitations are required to determine the time constants. If the reference frequency is 1 MHz, then N is 2,400 to 2,483. The VCO is required to cover a tuning range of 83.5 MHz, but there must be additional tuning range to allow for process variations. A tuning range of 10%, or about 250 MHz, seems necessary, as ±5% seems quite possible. Thus, a VCO constant of 250 MHz/V, or about 1.6 Grad/s/V, can be expected, assuming a 2-V supply and some room on either side of the rails. Now the remaining loop components that need to be determined are the charge pump current I, the integrating capacitor value C_1, and the phase-lead correction resistor R. From (3.22), the ratio of C_1 and I can be determined:

$$\frac{C_1}{I} = \frac{K_{\text{VCO}}}{2\pi \cdot N\omega_n^2} = 1.08 \cdot 10^{-4}$$

If C_1 is chosen to be 5 nF, then the charge pump current is $I = 46.3\ \mu\text{A}$. Note that this capacitor could not be integrated, but loop filters are often realized off chip. Now, making use of (3.23), the loop filter resistor R can be determined as well:

$$R = 2\zeta\sqrt{\frac{2\pi \cdot N}{IK_{\text{VCO}}C_1}} = 2(0.707)\sqrt{\frac{2\pi \cdot 2{,}400}{46.3\ \mu\text{A} \cdot 1.6\,\dfrac{\text{Grad}}{s} \cdot 5\ \text{nF}}} = 9.02\ \text{k}\Omega$$

The reference at 1 MHz is two decades higher than the loop corner frequency of 10 kHz. If we assume a first-order roll off for the loop of 20 dB/dec, then the reference signal will be attenuated 40 dB by the loop. Additional filtering may also be achieved by adding $C_2 = 500$ pF.

3.6.2 Nonlinear Transient Behavior

When a PLL is first turned on, or if it experiences a large frequency step at the input, then it may lose lock. In this case, the linear control theory that has been used so far will not apply as nonlinearities are involved in lock acquisition. The main reason for nonlinearity is the finite linear range of the phase detector. Additionally, there is a finite range over which the loop can acquire lock because VCOs have a finite tuning range. If the loop attempts to lock the VCO to a frequency outside its range, then the loop will never acquire lock. In addition, if the loop has a finite dc gain, then the range of lock acquisition may also be limited by the finite range of the phase detector.

In general, a frequency step will result in a nonzero phase error. The general transfer function for phase error is

$$\frac{\theta_e}{\theta_R} = \frac{s}{s + KF(s)} \tag{3.55}$$

Now, if a frequency step is applied to this system, the steady-state phase error will be

$$\theta_{e\_ss} = \lim_{s \to 0} \left[\left(\frac{\Delta\omega}{s^2} \right) \cdot \left(\frac{s}{s + KF(s)} \right) \cdot s \right] = \frac{\Delta\omega}{KF(0)} \tag{3.56}$$

In the second-order PFD/CP system, the steady-state phase error will always be zero because there is an integrator in $F(s)$, and $F(0)$ will go to ∞ in (3.56). In other loops without an integrator in $F(s)$, the phase error will be finite. When the steady-state error exceeds the linear range of the phase detector, the loop will lose lock. So, for instance, assume that a loop without an integrator uses an XOR phase detector, and the nominal phase error at ω_{nom} is $\pi/4$. Then the maximum phase error that can be tolerated is an additional $\pi/4$. Thus,

$$\Delta\omega_{\max} = \pm KF(0)\frac{\pi}{4} \tag{3.57}$$

However, in the case of the PDF/CP loop, this is not an issue as, in lock, the steady-state phase error is always zero. In this case, the locking range is determined exclusively by the VCO.

So, if the loop is going to experience a transient frequency step, then how long does it take the loop to reacquire lock? In this situation, the loop goes into a frequency-acquisition mode, and the output of the PFD (in frequency-detection mode) will look something like that shown in Figure 3.9. In this case, the charge pump will put out pulses of current of value I, which vary in width between almost a complete reference period and almost zero. Therefore, the average current produced by the charge pump until the loop acquires lock will be approximately $I/2$. If it is assumed that all of this current flows onto the capacitor C_1, then the change in voltage across the capacitor as a function of time will be

$$\frac{\Delta v_C}{\Delta t} = \frac{I}{2C_1} \tag{3.58}$$

Therefore, the settling time will be

$$T_s = \frac{2\Delta v_C C_1}{I} \tag{3.59}$$

From this equation, and making use of the relationship in (3.3) and (3.20), the settling time can be determined for an input frequency change $\Delta\omega$ as

$$T_s = \frac{2C_1\Delta\omega N}{IK_{\text{VCO}}} = \frac{\Delta\omega}{\pi\omega_n^2} \tag{3.60}$$

It should be noted that Δv_C and K_{VCO} relate to the change in VCO frequency, not to the change in input frequency. This leads to a factor of N in the equation. Thus, one can see directly that, as the loop bandwidth expands, the settling time will decrease as expected. Even though this is the main result that we are interested in, a few more details about the transient behavior of the control voltage are of interest. To start this discussion, we will assume that the loop filter does not have a capacitor C_2 and is currently charging up towards lock as shown in Figure 3.21.

In this case, the charge pump current will alternately turn on and off. When the charge pump current is off, then v_R will be zero, and the control voltage v_c will be equal to the voltage across the capacitor v_{C1}. However, when the charge pump current is on, then v_c will be equal to $v_c = v_{C1} + IR$, where IR is the voltage drop across the resistor. This is illustrated in Figure 3.22.

When the capacitor C_2 is included, as a result of its filtering effect, the behavior is a little more complicated. In this case, when the charge pump is on, most of the current still flows into C_1, which still happily charges towards lock, but when it turns off, there is no longer an instantaneous change in v_c. In this case, C_2 keeps v_c high, and current flows from C_2 back into C_1 through R. This is shown in Figure 3.23.

Thus, the presence of C_2 tends to smooth out the ripple on the control voltage. For context, the same plot as the one in Figure 3.22 is shown in Figure 3.24(b), where voltage waveforms are also plotted over a larger percentage of acquisition shown in Figure 3.24(a).

Example 3.4: Simulation and Estimation of Loop Settling Times

A 3.7–4.3-GHz synthesizer with a step size of 1 MHz is required. A 40-MHz crystal oscillator, a charge pump with $2\pi \cdot 100$ μA output current, and a VCO

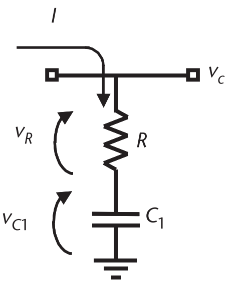


Figure 3.21 Simplified loop filter to illustrate settling behavior.

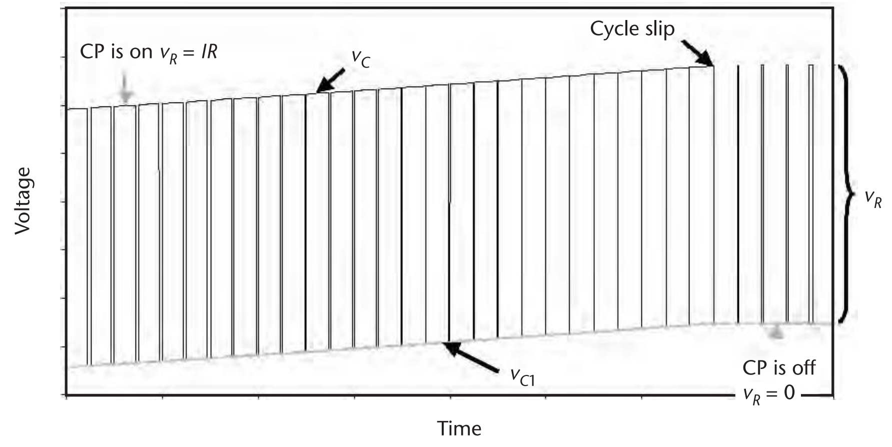


Figure 3.22 Example showing the voltages on the loop filter during acquisition (no C_2 present).

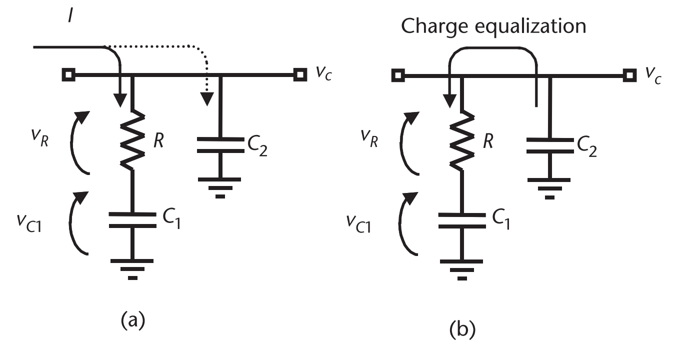


Figure 3.23 Illustration of loop filter behavior (C_2 present): (a) charge pump is on and charging both C_1 and C_2, and (b) charge pump is off, and C_2 is discharging into C_1.

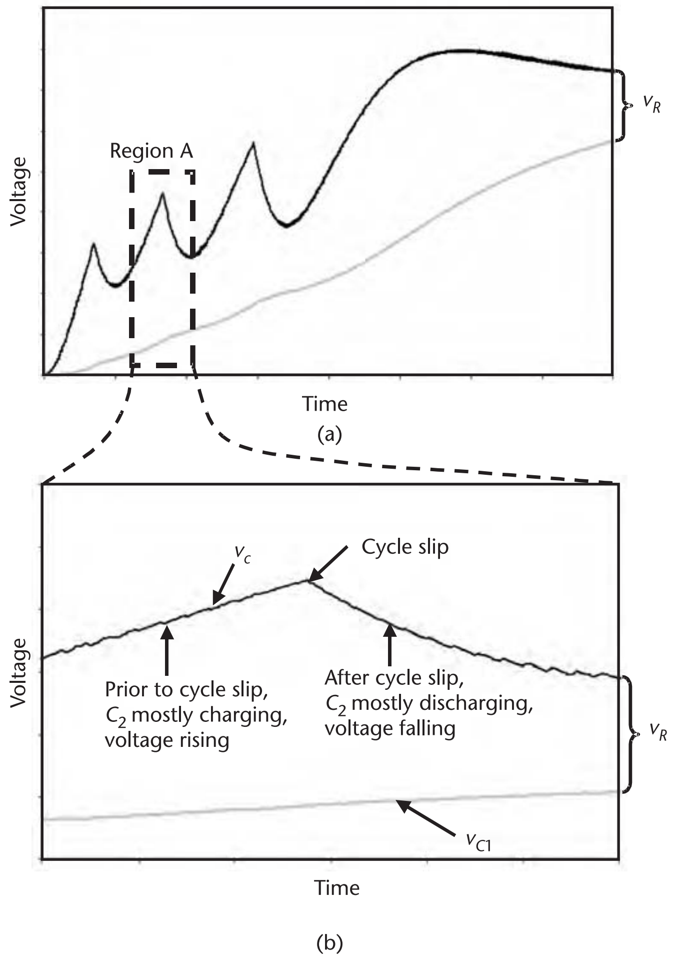


Figure 3.24 Example showing the voltages on the loop filter during acquisition (C_2 present): (a) complete settling, and (b) zoom in on region A.

(operating from a 3V supply) are available. Design a fractional-N synthesizer with a loop bandwidth of 150 kHz using these components. Estimate the settling time of the loop for a 30-MHz and 300-MHz frequency step. Simulate and compare.

Solution: First, if the VCO is operating with a 3V supply and must have a 600-MHz tuning range, we can estimate that its K_{VCO} will be 200 MHz/V. In addition, since we know the charge pump current, we know that the K_{phase} will be 100 μA/rad. For a VCO with a nominal frequency of 4 GHz and a reference frequency of 40 MHz, the division ratio will be 100. The next step is to size the loop filter. A 3-dB frequency of 150 kHz requires a natural frequency of 75 kHz (see Example 3.2). Thus, components can be determined as

$$C_1 = \frac{IK_{\text{VCO}}}{2\pi \cdot N\omega_n^2} = \frac{2\pi \cdot 100\ \mu\text{A} \cdot \left(2\pi \cdot 200\dfrac{\text{MHz}}{V}\right)}{2\pi \cdot 100(2\pi \cdot 75\ \text{kHz})^2} = 5.66\ \text{nF}$$

In order to set R, we need to pick a damping constant. Let us pick $1/\sqrt{2}$, or 0.707, which is a popular choice. Now,

$$\begin{aligned} R &= 2\zeta\sqrt{\frac{2\pi \cdot N}{IK_{\text{VCO}}C_1}} \\ &= 2\left(\frac{1}{\sqrt{2}}\right)\sqrt{\frac{2\pi \cdot 100}{2\pi \cdot 100\ \mu\text{A}\left(2\pi \cdot 200\dfrac{\text{MHz}}{V}\right) \cdot 5.66\ \text{nF}}} \\ &= 530\Omega \end{aligned}$$

and we will set $C_2 = 566$ pF at one-tenth the value of C_1.

Now, a step in output frequency of 30 MHz and 300 MHz corresponds to a step in the reference frequency of 0.3 MHz and 3 MHz, respectively. We learned in Example 3.2 that the maximum input frequency step that can be tolerated for a system with these parameters is 1 MHz. Therefore, the first frequency step will be a linear one, and the output will follow the theory of the previous section. Therefore, we expect that it will take approximately 15 μs to settle, as we discovered previously.

In contrast, the second frequency step will involve cycle slipping. For this nonlinear case, we use the formula given in (3.60) to estimate the acquisition time as

$$T_s = \frac{\Delta\omega}{\pi\omega_n^2} = \frac{2\pi \cdot 3\ \text{MHz}}{\pi(2\pi \cdot 75\ \text{kHz})^2} = 27\ \mu\text{s}$$

Therefore, complete settling in this case should take 27 μs, plus an additional 15 μs for phase lock.

This behavior can be simulated using ideal components in a simulator such as Cadence's Spectre. The blocks for the divider, VCO, PFD, and charge pump can be programmed using ideal behavioral models. These can be connected to the loop filter that has just been designed. From these simulations, we can look at the control voltage on the VCO to verify the performance of the loop. A plot of the response of the system to a 0.3-MHz step at its input, compared to simple theory, is shown in Figure 3.25. From this graph, it is easy to see that the simple theory does an excellent job of predicting the settling behavior of the loop with only a slight deviation. This small discrepancy is most likely due to the presence of C_2 and to the sampling nature of the loop components.

The second frequency step can be simulated as well. The results of this simulation are plotted in Figure 3.26 and compared to the linear voltage ramp suggested by simple theory previously. In this case, this plot shows that the nonlinear response is predicted fairly well by the simple formula; however, the actual response is slightly faster. The tail of this graph is cut off, but the loop settled in about 39 μs, which is very close to the 42 μs predicted. The main difference between the simple estimate and reality is the fact that phase acquisition begins before the PLL actually reaches its final frequency value. We predicted it would begin in this simulation at 30 μs (when the simple theory predicts that the voltage ramp will reach its final value), but the linear portion of the graph actually starts earlier than this at about 25 or 26 μs. This accounts for our slightly pessimistic estimate. Still, such a simple estimate is remarkably good at predicting quite complicated behavior.

3.7 Phase Noise and Timing Jitter in PLL Synthesis

Noise in synthesizers comes from all the different circuits and components that make up the control loop. Synthesizer noise performance is usually classified in terms of phase noise, which is a measure of how much the output diverges from

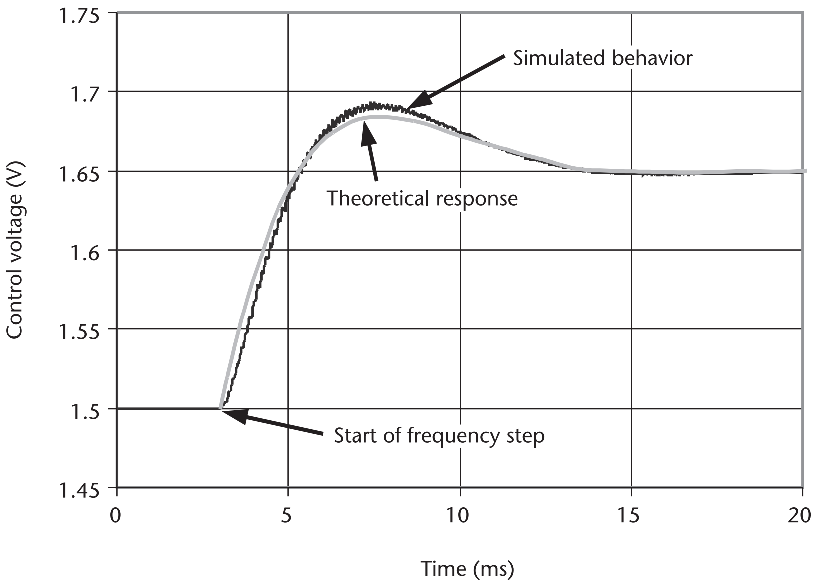


Figure 3.25 Response of the PLL design's control voltage during a 30-MHz frequency step.

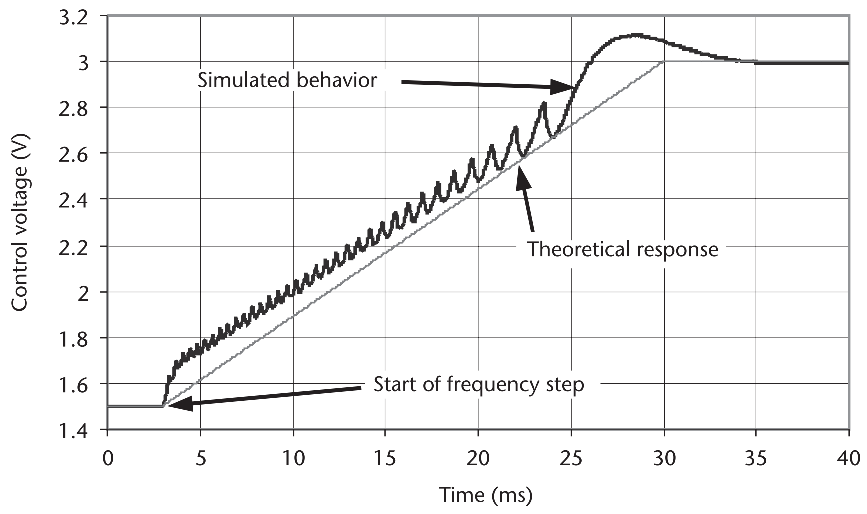


Figure 3.26 Response of the PLL design's control voltage during a 300-MHz frequency step.

an ideal impulse function in the frequency domain. We are primarily concerned with noise that causes fluctuations in the phase of the output, rather than noise that causes amplitude fluctuations in the tone, since the output typically has a fixed, limited amplitude. The output signal of a synthesizer can be described as

$$v_{\text{out}}(t) = V_0 \cos[\omega_{\text{LO}} t + \varphi_n(t)] \tag{3.61}$$

Here, $\omega_{\text{LO}} t$ is the desired phase of the output and $\varphi_n(t)$ are random fluctuations in the phase of the output due to any one of a number of sources. Phase noise is often quoted in units of dBc/Hz, while timing jitter is often quoted in units of picoseconds.

The phase-fluctuation term $\varphi_n(t)$ may be random phase noise or discrete spurious tones, as shown in Figure 3.27. The discrete spurs at a synthesizer output are most likely due to the fractional-N mechanism (discussed in detail in Chapter 9) and the phase noise in an oscillator, which is mainly due to thermal, flicker, or $1/f$ noise and the finite Q of the oscillator tank.

The phase fluctuation is assumed to have a sinusoidal form as

$$\varphi_n(t) = \varphi_p \sin(\omega_m t) \tag{3.62}$$

where φ_p is the peak phase fluctuation, and ω_m is the offset frequency from the carrier. Substituting (3.62) into (3.61) gives

$$\begin{aligned} v_{\text{out}}(t) &= V_0 \cos[\omega_{\text{LO}} t + \varphi_p \sin(\omega_m t)] \qquad (3.63) \\ &= V_0\{\cos(\omega_{\text{LO}} t) \cos[\varphi_p \sin(\omega_m t)] - \sin(\omega_{\text{LO}} t) \sin[\varphi_p \sin(\omega_m t)]\} \end{aligned}$$

For a small phase fluctuation, the above equation can be simplified as

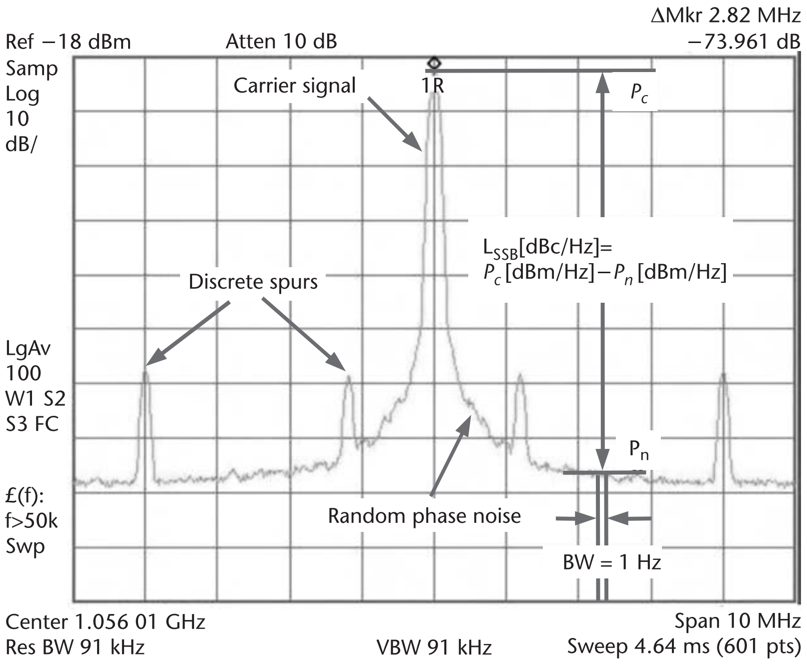


Figure 3.27 Example of phase noise and spurs observed using a spectrum analyzer.

$$
\begin{aligned}
v_0(t) &= V_0[\cos(\omega_{\text{LO}}t) - \varphi_p \sin(\omega_m t)\sin(\omega_{\text{LO}}t)] \\
&= V_0\left\{\cos(\omega_{\text{LO}}t) - \frac{\varphi_p}{2}[\cos(\omega_{\text{LO}} - \omega_m)t - \cos(\omega_{\text{LO}} + \omega_m)t]\right\}
\end{aligned}
\tag{3.64}
$$

It is now evident that the phase-modulated signal includes the carrier signal tone and two symmetric sidebands at any offset frequency, as shown in Figure 3.27. A spectrum analyzer measures the phase noise power in dBm/Hz, but often phase noise is reported relative to the carrier power as

$$
\varphi_n^2(\Delta\omega) = \frac{\text{Noise}(\omega_{\text{LO}} + \Delta\omega)}{P_{\text{carrier}}(\omega_{\text{LO}})}
\tag{3.65}
$$

where Noise is the noise power in a 1-Hz bandwidth, and P_{carrier} is the power of the carrier or LO tone at the frequency at which the synthesizer is operating. In this form, phase noise has units of rad^2/Hz. Often this is quoted as so many decibels down from the carrier, or in dBc/Hz. To further complicate this, both single-sideband (SSB) and double-sideband phase noise can be defined. SSB phase noise is defined as the ratio of power in one sideband per hertz of bandwidth, at an offset $\Delta\omega$ away from the carrier, to the total signal power. The SSB phase noise power spectral density (PSD) to carrier ratio, in units of [dBc/Hz], is defined as

$$
PN_{\text{SSB}}(\Delta\omega) = 10\log\left[\frac{\text{Noise}(\omega_{\text{LO}} + \Delta\omega)}{P_{\text{carrier}}(\omega_{\text{LO}})}\right]
\tag{3.66}
$$

Combining (3.64) and (3.66), this equation can be rewritten as

$$PN_{\text{SSB}}(\Delta\omega) = 10\log\left[\frac{\frac{1}{2}\left(\frac{V_0\varphi_p}{2}\right)^2}{\frac{1}{2}V_0^2}\right] = 10\log\left(\frac{\varphi_p^2}{4}\right) = 10\log\left(\frac{\varphi_{\text{rms}}^2}{2}\right) \tag{3.67}$$

where φ_{rms}^2 is the root mean squared phase noise PSD in units of [rad^2/Hz]. Note that SSB phase noise is by far the most common type reported, and, often, it is not specified as SSB but simply reported as phase noise. However, alternatively, double-sideband phase noise is given by

$$PN_{\text{DSB}}(\Delta\omega) = 10\log\left[\frac{\text{Noise}(\omega_{\text{LO}}+\Delta\omega)+\text{Noise}(\omega_{\text{LO}}-\Delta\omega)}{P_{\text{carrier}}(\omega_{\text{LO}})}\right] = 10\log(\varphi_{\text{rms}}^2) \tag{3.68}$$

From either the SSB or double-sideband phase noise, the rms jitter can be obtained as

$$\varphi_{\text{rms}}(\Delta f) = \frac{180}{\pi}\sqrt{10^{\frac{PN_{\text{DSB}}(\Delta f)}{10}}} = \frac{180\sqrt{2}}{\pi}\sqrt{10^{\frac{PN_{\text{SSB}}(\Delta f)}{10}}}\ \left(\text{deg}/\sqrt{\text{Hz}}\right) \tag{3.69}$$

It is also quite common to quote integrated phase jitter. The rms integrated phase noise of a synthesizer is given by

$$\text{IntPN}_{\text{rms}} = \sqrt{\int_{\Delta f_1}^{\Delta f_2}\varphi_{\text{rms}}^2(f)\,df} \tag{3.70}$$

The limits of integration are usually the offsets corresponding to the lower and upper frequencies of the bandwidth of the information being transmitted.

In addition, it should be noted that dividing or multiplying a signal in the time domain also multiplies or divides the phase noise. Thus, if a signal is translated in frequency by a factor of N, then the phase noise is related by

$$\varphi_{\text{rms}}^2(N\omega_{\text{LO}}+\Delta\omega) = N^2\cdot\varphi_{\text{rms}}^2(\omega_{\text{LO}}+\Delta\omega) \tag{3.71}$$

$$\varphi_{\text{rms}}^2\left(\frac{\omega_{\text{LO}}}{N}+\Delta\omega\right) = \frac{\varphi_{\text{rms}}^2(\omega_{\text{LO}}+\Delta\omega)}{N^2}$$

Note that this assumes that the circuit that did the frequency translation is noiseless. Also note that the phase noise is scaled by N^2 rather than N because we are dealing with noise in units of voltage squared rather than noise voltage.

One should also be careful when measuring noise on a spectrum analyzer. In a spectrum analyzer, the noise power is normally not integrated over a 1-Hz bandwidth. As shown in Figure 3.28, the spectrum analyzer downconverts the input signal to baseband through two filters, an analog IF filter and a digital video filter. Phase noise reading is thus dependent on the resolution bandwidth (RBW), which is the IF filter bandwidth, in a spectrum analyzer. For example, a phase noise reading of –100 dBc with an RBW of 10 Hz in a spectrum analyzer would be –90 dBc with another RBW setting of 100 Hz. Video bandwidth (VBW), which is the video filter bandwidth, smoothes noise for easier identification and measurement of low-level signals. The best sensitivity in a spectrum analyzer is achieved by using the narrowest RBW, minimum RF attenuation, and sufficient video filtering to smooth the noise (normally, VBW is less than 1% of the RBW). Since a spectrum analyzer measures the baseband signal power, the PSD reading is normally SSB. A proper sweeping setting is required to obtain symmetric sideband measurement. The penalty for sweeping too fast is an uncalibrated display of sidebands.

3.7.1 Various Noise Sources in PLL Synthesizers

3.7.1.1 Origin of Noise

There are many sources of noise. Noise in resistors is generated by thermal energy causing random electron motion. Noise in active devices can be thermal channel noise, 1/*f* noise, or shot noise, and is further discussed in Appendix B. Noise can also be coupled into the circuit under test through electromagnetic coupling from other circuits or external noise sources, such as microwave ovens, cell phones, or pagers. Noise can also be injected through the substrate due to a combination of capacitive and conductive coupling from other circuits on the same die, for example, other oscillators, power amplifiers, or digital circuits [8].

3.7.1.2 VCO Noise

All the circuits in the synthesizer contribute to the overall noise in different ways, and the noise they produce has different characteristics. For instance, the phase noise from a VCO can be described as [9]

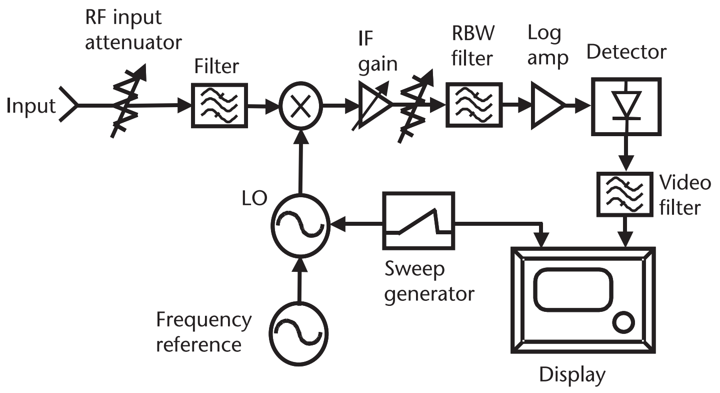


Figure 3.28 Spectrum analyzer basic block diagram.

$$\varphi_{\text{VCO}}^2(\Delta\omega) = \frac{C}{\Delta\omega^2} + D \tag{3.72}$$

where C is a constant of proportionality, and ω is the frequency offset from the carrier. Thus, at most frequencies of interest, the phase noise produced by the VCO will go down at 20 dB/dec as we move away from the carrier. This will not continue indefinitely, as thermal noise will put a lower limit on this phase noise D, which, for most integrated VCOs, is somewhere between −120 and −150 dBc/Hz. VCO phase noise is usually dominant outside the loop bandwidth and of less importance at low offset frequencies.

3.7.1.3 Crystal Reference Noise

Crystal resonators are widely used in frequency-control applications because of their unequaled combination of high Q, stability, and small size. The resonators are classified according to "cut," which is the orientation of the crystal wafer (usually made from quartz) with respect to the crystallographic axes of the material. More will be said about crystal oscillators in Chapter 8. The total noise PSD of a crystal oscillator can be found by Leeson's formula [10]:

$$\varphi_{XTAL}^2(\Delta\omega) = 10^{-16\pm1} \cdot \left[1 + \left(\frac{\omega_0}{2\Delta\omega \cdot Q_L}\right)^2\right]\left(1 + \frac{\omega_c}{\Delta\omega}\right) \tag{3.73}$$

where ω_0 is the oscillator output frequency; ω_c is the corner frequency between $1/f$ and thermal noise regions, which is normally in the range 1~10 kHz; and Q_L is the loaded quality factor of the resonator. Since Q_L for crystal resonator is very large (normally in the order of 10^4 to 10^6), the reference noise contributes only to the very close-in noise, and it quickly reaches the thermal noise floor at an offset frequency of around ω_c.

3.7.1.4 Frequency-Divider Noise

Frequency dividers consist of switching logic circuits, which are sensitive to clock timing jitter. The jitter in the time domain can be converted to phase noise in the frequency domain. Time jitter or phase noise occurs when rising and falling edges of digital dividers are superimposed with spurious signals, such as thermal noise and flicker noise in semiconductor materials. Ambient effects result in variation of the triggering level due to temperature and humidity. Frequency dividers generate spurious noise especially for high-frequency operation. Dividers do not generate signals; rather, they simply change their frequency. Kroupa provided an empirical formula that describes the amount of phase noise that frequency dividers add to a signal [7, 11]:

$$\varphi_{\text{Div\_Added}}^2(\Delta\omega) \approx \frac{10^{-14\pm1} + 10^{-27\pm1}\omega_{\text{do}}^2}{2\pi \cdot \Delta\omega} + 10^{-16\pm1} + \frac{10^{-22\pm1}\omega_{\text{do}}}{2\pi} \tag{3.74}$$

where ω_{do} is the divider output frequency, and $\Delta\omega$ is the offset frequency. Notice that the first term in (3.74) represents the flicker noise and the second term gives the white thermal-noise floor. The third term is caused by timing jitter due to coupling and ambient and supply variations.

3.7.1.5 Phase Detector Noise

Phase detectors experience both flicker and thermal noise. At large offsets, phase detectors generate a white phase noise floor of about −160 dBc/Hz, which is thermal noise dominant. The noise PSD of phase detectors is given by [12]

$$\varphi_{\text{PD}}^2(\Delta\omega) \approx \frac{2\pi \cdot 10^{-14\pm 1}}{\Delta\omega} + 10^{-16\pm 1} \tag{3.75}$$

3.7.1.6 Charge Pump Noise

The noise of the charge pump can be characterized as an output noise current and is usually given in pA/$\sqrt{\text{Hz}}$. Note that, at this point in the loop, the phase is represented by the current. The charge pump output current noise can be a strong function of the frequency and of the width of the current pulses; therefore, for low noise operation, it is desirable to keep the charge pump currents matched as well as possible. This is because current sources only produce noise when they are on. In an ideal loop, when locked, the charge pump is always off. However, nonidealities result in finite pulses, but the closer reality comes to matching the ideal case, the less noise will be produced. Also note that, as the frequency is decreased, 1/*f* noise will become more important, causing the noise to increase. This noise can often be the dominant noise source at low-frequency offsets. Charge pump noise can be simulated with proper tools and the results depend on the design in question, so no simple formula will be given here. More will be said on this topic in Chapter 7.

3.7.1.7 Loop Filter Noise

Loop filters are simple resistor-capacitor circuits and can be analyzed for noise in the frequency domain in a linear manner. The most common loop filter that has been examined in this chapter will now be analyzed. It contains only one noise source, the thermal noise associated with R. Thus, the loop filter with associated noise can be drawn as shown in Figure 3.29. Now, the noise voltage develops a current flowing through the series combination of C_1, C_2, and R (assuming that the charge pump and VCO are both open circuits), which is given by

$$i_{n\_\text{LPF}} = \frac{1}{R} \cdot \frac{v_n s}{s + \dfrac{C_1 + C_2}{C_1 C_2 R}} \approx \frac{1}{R} \cdot \frac{v_n s}{s + \dfrac{1}{C_2 R}} \tag{3.76}$$

Thus, this noise current will have a highpass characteristic; therefore, the loop will not produce any noise at dc, and this noise will increase with frequency until

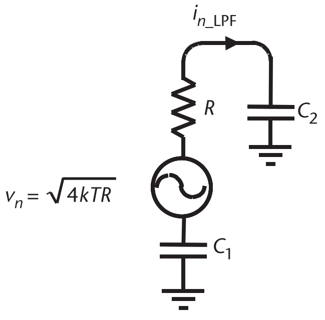


Figure 3.29 Loop filter with thermal noise added.

the highpass corner is reached, after which point it will be flat. Other filters can be analyzed as well and will be looked at in more detail in Chapter 7.

3.7.2 In-Band and Out-of-Band Phase Noise in PLL Synthesis

The noise transfer functions for the various noise sources in the loop can be derived quite easily using the theory already presented. There are two noise transfer functions: one for the VCO and one for all other sources of noise in the loop. All noise generated by the PFD, charge pump, divider, and loop filter is referred back to the input, and the noise from the VCO is referred to the output, as shown in Figure 3.30. The transfer function for $\varphi_{\text{noise I}}(s)$ has already been derived (as it is the same as the loop transfer function in the continuous domain) and is given by

$$\frac{\varphi_{\text{noise out}}(s)}{\varphi_{\text{noise I}}(s)} = \frac{F(s) K_{\text{VCO}} K_{\text{phase}}}{s + \dfrac{F(s) K_{\text{VCO}} K_{\text{phase}}}{N}} \tag{3.77}$$

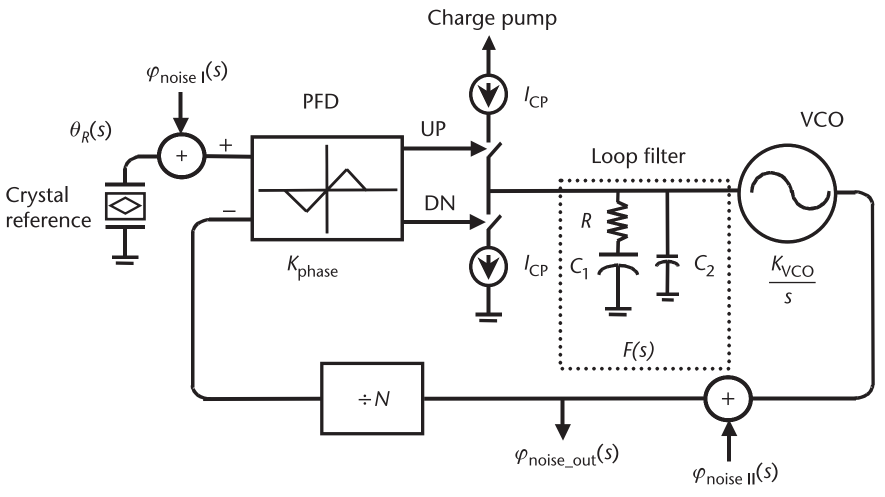


Figure 3.30 A synthesizer showing places where noise is injected.

where for the PFD/CP loop, the transfer function for the filter, divider, and crystal reference noise becomes

$$\frac{\varphi_{\text{noise out}}(s)}{\varphi_{\text{noise I}}(s)} = \frac{\dfrac{IK_{\text{VCO}}}{2\pi \cdot C_1}(1 + RC_1 s)}{s^2 + \dfrac{IK_{\text{VCO}}}{2\pi \cdot N} Rs + \dfrac{IK_{\text{VCO}}}{2\pi \cdot NC_1}} \tag{3.78}$$

This is a lowpass function, which means that, for low frequencies (inside the loop bandwidth), the loop will track the input phase (which includes the phase noise); thus, this noise will be transferred to the output. At higher offset frequencies (outside the loop bandwidth), this noise is suppressed as the loop prevents the VCO from following these changes in phase. Also note that the division ratio plays a very important part in this transfer function. It can be seen that, at low reference frequencies, where the s and s^2 terms in (3.78) can be ignored relative to the constant terms, a higher division ratio N directly results in higher phase noise. This is one of the strongest arguments for using fractional-N architectures in synthesizers, as, with large division ratios, it is hard to get low phase noise performance.

The transfer function for the noise due to the VCO is slightly different. In this case, input noise is set to zero; then, the transfer function is derived in the usual way. It is given in general by

$$\frac{\varphi_{\text{noise out}}(s)}{\varphi_{\text{noise II}}(s)} = \frac{s}{s + \dfrac{F(s) K_{\text{VCO}} K_{\text{phase}}}{N}} \tag{3.79}$$

Using our loop will give

$$\frac{\varphi_{\text{noise out}}(s)}{\varphi_{\text{noise II}}(s)} = \frac{s^2}{s^2 + \dfrac{IK_{\text{VCO}}}{2\pi \cdot N} Rs + \dfrac{IK_{\text{VCO}}}{2\pi \cdot NC_1}} \tag{3.80}$$

This is a highpass filter. Thus, at low offsets inside the loop bandwidth, the VCO noise is suppressed by the feedback action of the loop. Outside the loop bandwidth, however, the VCO is essentially free running; thus, the loop noise approaches the VCO noise.

Example 3.5: System Phase Noise Calculation

Estimate the phase noise for the synthesizer designed in Example 3.4. The VCO has a phase noise of −120 dBc/Hz at a 1-MHz offset (it bottoms out at −130 dBc/Hz), and the charge pump puts out a noise current of 10 pA/$\sqrt{\text{Hz}}$. Ignoring PFD, divider, and reference noise sources, plot the phase noise. In addition, what would the phase noise of an equivalent integer-N design be?

Solution: From Example 3.4, we know the charge pump current, and we know that the K_{phase} will be 100 μA/rad. Now, in the case of the integer-N synthesizer, the reference must be 1 MHz in order to get a step size of 1 MHz. Therefore, the division ratio will be 4,000. Knowing that we want a loop bandwidth of 150 kHz means that we need a natural frequency of 75 kHz (assuming a damping constant of 0.707), and this means that, for the integer-N design, C_1 and R are 141.5 pF and 21.2 kΩ respectively. Now, we will assume that the VCO follows the 20 dB/dec rule just outlined. Therefore, we can come up with a linear expression for the phase noise of the VCO based on (3.72):

$$C = \log^{-1}\left(\frac{PN_{\text{VCO}}}{10}\right) \cdot \Delta\omega^2 = \log^{-1}\left(\frac{-120}{10}\right) \cdot (2\pi \cdot 1 \text{ MHz})^2 = 39.5 \frac{\text{rad}^4}{\text{Hz}^2}$$

Since the VCO bottoms out at −130 dBc/Hz, we can determine the D term of the VCO phase noise equation (3.72):

$$D = \log^{-1}\left(\frac{PN_{\text{VCO}}}{10}\right) = \log^{-1}\left(\frac{-130}{10}\right) = 10^{-13} \frac{\text{rad}^2}{\text{Hz}}$$

This can be turned into an equation that has units of voltage instead of units of voltage squared:

$$\varphi_{\text{VCO}}(\Delta\omega) = \sqrt{\frac{39.5}{\Delta\omega^2} + 10^{-13}} \frac{\text{rad}}{\sqrt{\text{Hz}}}$$

The output noise current from the charge pump can be input-referred by dividing by K_{phase}:

$$\text{Noise}_{\text{CP}} = \frac{i_n}{K_{\text{phase}}} = \frac{10 \dfrac{pA}{\sqrt{\text{Hz}}}}{100 \dfrac{\mu A}{\text{rad}}} = 100n \cdot \frac{\text{rad}}{\sqrt{\text{Hz}}}$$

The noise from the loop filter must also be moved back to the input:

$$\text{Noise}_{\text{LPF}}(\omega) = \frac{1}{K_{\text{phase}}} \left| \frac{\sqrt{\dfrac{4kT}{R}}\, j\omega}{j\omega + \dfrac{1}{C_2 R}} \right|$$

Clearly, noise from the LPF is dependent on filter-component values as well as the phase detector gain. Now, input-referred noise from the loop filter and the charge pump can both be substituted into (3.78), while noise from the VCO can be substituted into (3.80) to determine the contribution to the phase noise at the output. Once each component value at the output is calculated, the overall noise can be computed (noting that noise adds as power). So, for instance, in the case of the noise due to the charge pump, the output phase noise for the fractional-*N* case is [from (3.78)]

$$\varphi_{\text{noise out\_CP}}(s) = \frac{2.22 \cdot 10^{13}(1 + 3 \cdot 10^{-6}\, s)}{s^2 + 6.66 \cdot 10^5 s + 2.22 \cdot 10^{11}} 100n\left(\frac{\text{rad}}{\sqrt{\text{Hz}}}\right)$$

Therefore, to plot phase noise in dBc/Hz, we take

$$PN_{\text{CP}}(\Delta\omega) = 20 \log\left[\left|\frac{2.22 \cdot 10^{13}(1 + 3 \cdot 10^{-6}\, j\Delta\omega)}{(j\Delta\omega)^2 + 6.66 \cdot 10^5\, j\Delta\omega + 2.22 \cdot 10^{11}}\right| 100n\right]\left(\frac{\text{rad}}{\sqrt{\text{Hz}}}\right)$$

The results of this calculation and similar ones for the other noise sources are shown in Figure 3.31. The total phase noise is computed by

$$\varphi_{\text{total}} = \sqrt{\varphi^2_{\text{noise out\_CP}} + \varphi^2_{\text{noise out\_VCO}} + \varphi^2_{\text{noise out\_LPF}}}$$

and is shown in the figure.

If we assume that the numbers given so far have been for SSB phase noise, then we can also compute the integrated phase noise of this design as

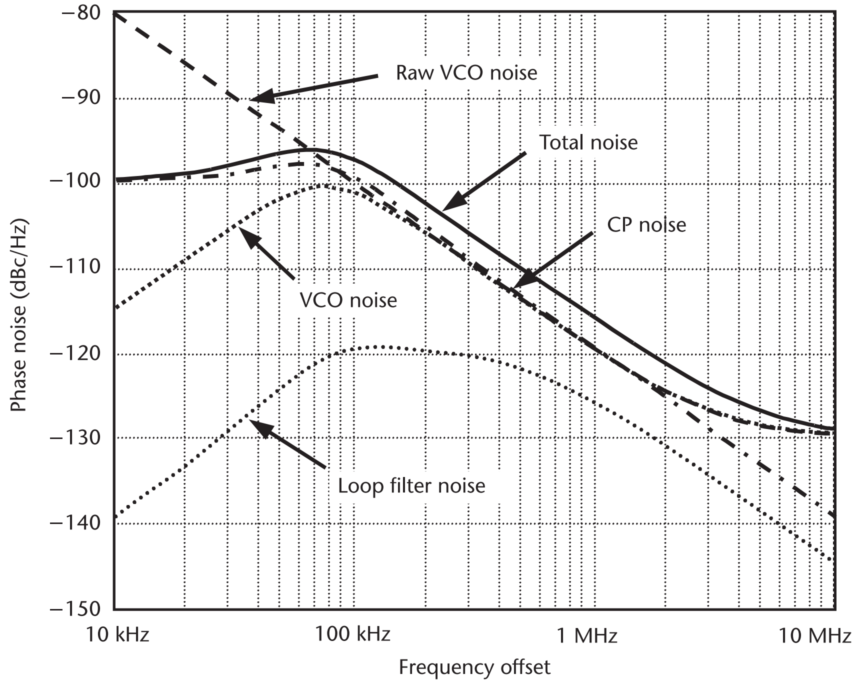


Figure 3.31 Phase noise of various components and overall phase noise for the system with fractional-*N* divider.

$$\mathrm{IntPN}_{\mathrm{rms}} = \frac{180\sqrt{2}}{\pi}\sqrt{\int_{f=10\,\mathrm{kHz}}^{f=10\,\mathrm{MHz}} \varphi_{\mathrm{total}}^{2}\, df} = 0.41^{\circ}$$

Note that, in this example, the loop filter noise is quite low and could have been ignored safely. Also note that, due to the frequency response of the filter even in-band, noise from the loop filter is attenuated at lower frequencies. Inside of the loop bandwidth, the total noise is dominated by the charge pump. Note that, out of band, the noise is slightly higher than the VCO noise. This is because the charge pump is still contributing. This can be corrected by making the loop bandwidth slightly smaller and, thus, attenuating the out-of-band contribution of the charge pump by a few more decibels.

With the integer-N numbers, the phase noise is shown in Figure 3.32. Note that with integer N, the noise is completely dominated by the charge pump, both inside and outside of the loop bandwidth. In order to reduce the effect of charge pump noise, the loop bandwidth in this case should be reduced by at least two orders of magnitude. Note also the dramatic change in the in-band phase noise performance between the two designs. While the fractional design has –100 dBc/Hz of in-band noise, this design has a performance of only –67 dBc/Hz. Note that the two numbers are different by 20 log(40), which is the ratio of the two divider values, as would be expected.

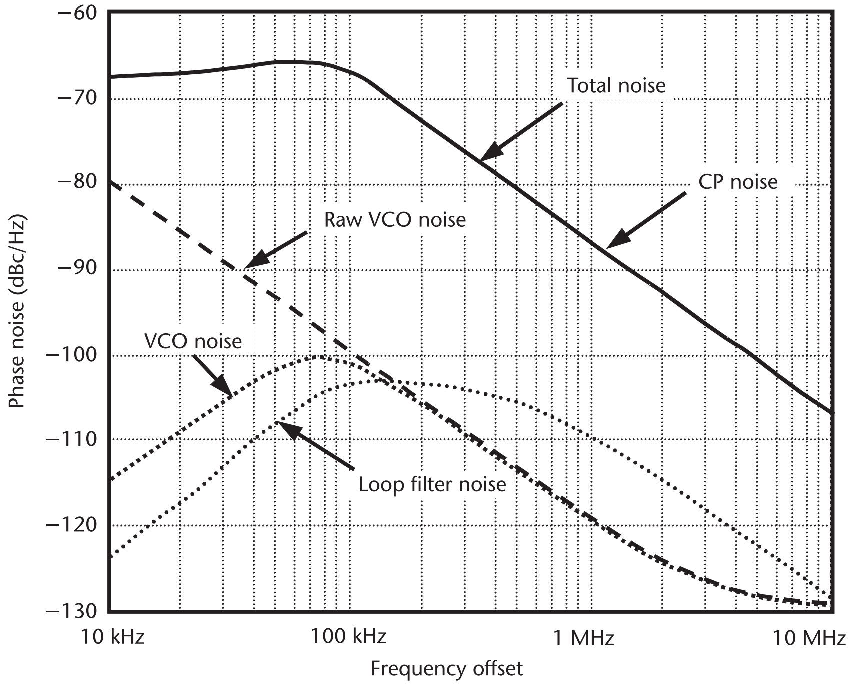


Figure 3.32 Phase noise of various components and overall phase noise for the system with integer-*N* divider.

References

[1] Best, R. E., *Phase-Locked Loops: Theory, Design, and Applications*, New York: McGraw-Hill, 1984.

[2] Blanchard, A., *Phase-Locked Loops: Applications to Coherent Receiver Design*, New York: John Wiley & Sons, 1976.

[3] Wolaver, D. H., *Phase-Locked Loop Circuit Design*, Upper Saddle River, NJ: Prentice Hall, 1991.

[4] Razavi, B., *Monolithic Phase-Locked Loops and Clock Recovery Circuits*, New York: Wiley-IEEE Press, 1996.

[5] Crawford, J. A., *Frequency Synthesizer Design Handbook*, Norwood, MA: Artech House, 1994.

[6] Gardner, F. M., *Phaselock Techniques*, New York: John Wiley & Sons, 1979.

[7] Egan, W. F., *Frequency Synthesis by Phase Lock*, New York: John Wiley & Sons, 2000.

[8] Amaya, R. E., et al., "EM and Substrate Coupling in Silicon RFICs," *IEEE J. Solid-State Circuits*, Vol. 40, No. 9, September 2005, pp. 1968–1971.

[9] Leeson, D. B., "A Simple Model of Feedback Oscillator Noise Spectrum," *Proc. IEEE*, February 1966, pp. 329–330.

[10] Watanabe, Y., et al., "Phase Noise Measurements in Dual-Mode SC-Cut Crystal Oscillators," *IEEE Trans. on Ultrasonics, Ferroelectrics and Frequency Control*, Vol. 47, No. 2, March 2000, pp. 374–378.

[11] Kroupa, V. F., "Noise Properties of PLL Systems," *IEEE Trans. on Communications*, Vol. 30, October 1982, pp. 2244–2252.

[12] Kroupa, V. F., "Jitter and Phase Noise in Frequency Dividers," *IEEE Trans. on Instrumentation and Measurement*, Vol. 50, No. 5, October 2001, p. 1241.

CHAPTER 4

Introduction to Digital IC Design

4.1 Digital Design Methodology and Flow

Electronic signals can be categorized as *analog* and *digital*. An analog signal is a waveform with continuous values, while a digital signal is a waveform with discrete values represented only by 1s and 0s. The performance of analog circuits can be evaluated by SPICE-like simulations, where SPICE is a simulation program with integrated circuit emphasis. The performance of digital circuits can be assessed using static timing analysis. Unlike the analog IC designs described in other chapters in this book, digital IC designs follow a different methodology and design flow. Digital design normally involves complicated logic functions with relatively low operating frequencies as compared to its analog counterpart. The most distinctive difference between digital and analog IC designs is that a complicated digital system can be designed using a hardware description language (HDL) such as Verilog or VHDL (VHSIC HDL, or very high-speed integrated circuit HDL), and a synthesis tool can automatically convert the HDL code into the gate-level representation of the system based on the timing requirements. Here, gates are taken to be simple combinatorial logic gates, such as inverters, AND, OR, NAND, NOR, XOR, XNOR, as shown in Figure 4.1, and simple storage gates, such as latches and flip-flops, which will be discussed in much further detail in Chapter 5. In contrast, analog circuits cannot be synthesized using HDL codes, although recent research has explored the feasibility of synthesizable analog circuits. For this reason, we still treat a high-speed frequency-divider circuit as an analog IC design, although it handles only binary logic signals, since this kind of high-speed circuit cannot be simply generated by logic synthesis. This chapter provides an overview of digital IC design methodology and an introduction to the Verilog HDL that has been widely used in digital IC designs. Note that VHDL is an alternative to Verilog and will not be discussed in detail in this book.

Analog IC design starts with system partitions and block specifications. Based on the system requirements, designers implement the analog blocks at the transistor level and perform SPICE-like simulations to verify the block performance. If the circuits meet the system requirements, the designers will layout the blocks based on the design rules specified by the vendor. Then, the parasitic capacitance and resistance will be extracted, and the circuits will be resimulated with the layout parasitic information. Adjustments to the circuits and layouts can be done iteratively until the final circuits and layouts meet all system specifications. In the analog design flow, almost every process is handcrafted, and no automated computer-

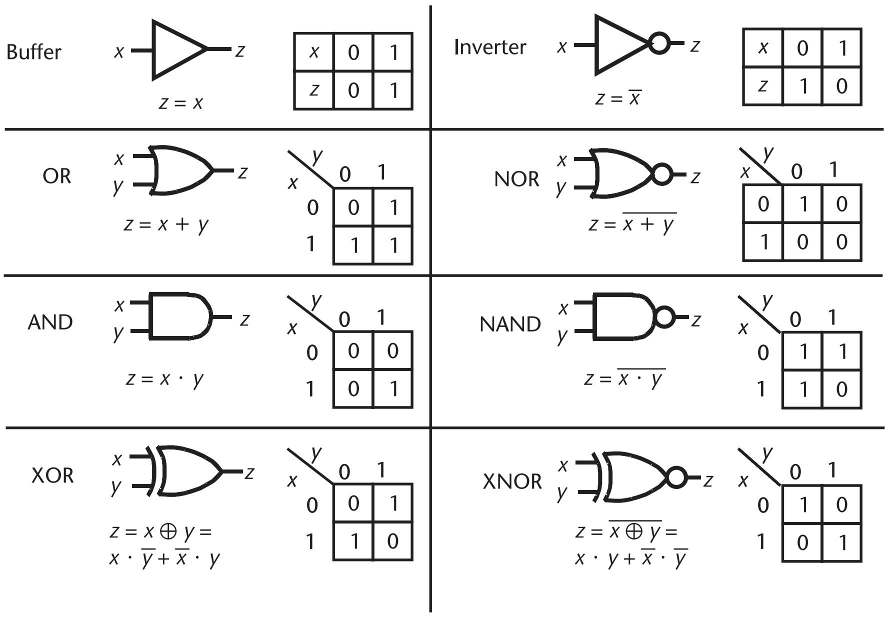

Buffer
$z = x$
Inverter
$z = \overline{x}$
OR
$z = x + y$
NOR
$z = \overline{x + y}$
AND
$z = x \cdot y$
NAND
$z = \overline{x \cdot y}$
XOR
$z = x \oplus y = x \cdot \overline{y} + \overline{x} \cdot y$
XNOR
$z = \overline{x \oplus y} = x \cdot y + \overline{x} \cdot \overline{y}$

Figure 4.1 Combinatorial logic gates showing symbols, equations, and truth tables.

assisted design tools are typically used in RF circuit synthesis and layout. Thus, a successful high-speed analog IC requires designers with extensive experience.

In comparison, the major steps in a digital IC design can be automated [1]. As illustrated in Figure 4.2, a digital design flow starts with the system function definition and technology selection. In addition to system functional specifications, designers have to understand all the circuit constraints, such as clock rate, delays, drive capability, and block size. Next, the designers will model the system at the register transfer level (RTL) using an HDL such as Verilog or VHDL. In RTL notation, all operations are modeled as the data transfer from one register to another with the desired combinational functions inserted between registers. The RTL codes can be verified by simulation. In the RTL simulation, the designers need to develop test benches that apply stimuli to the system inputs and capture the system outputs for evaluation.

Digital synthesis refers to the automated generation of logic-gate-level circuits based on high-level descriptions. This process revolutionized IC designs and allowed complicated digital systems to be designed using HDL. The synthesis output is normally a gate-level netlist that connects the basic logic gates based on the system functions specified in the HDL code. The gate-level netlist is technology-dependent. It can be further used to generate the layout for IC fabrication or to program a prefabricated IC, such as a field-programmable gate array (FPGA) [2]. In addition to HDL codes, some synthesis tools can also take logic schematic diagrams or finite-state machine diagrams as the synthesis inputs. Based on the given states and the state transitions, the synthesis tool can define state variables, assign states, choose flip-flops to represent the states, and then generate a netlist to represent the desired state transitions. Logic minimization can also be performed during synthesis. The synthesis tool will choose proper logic cells and insert proper buffers

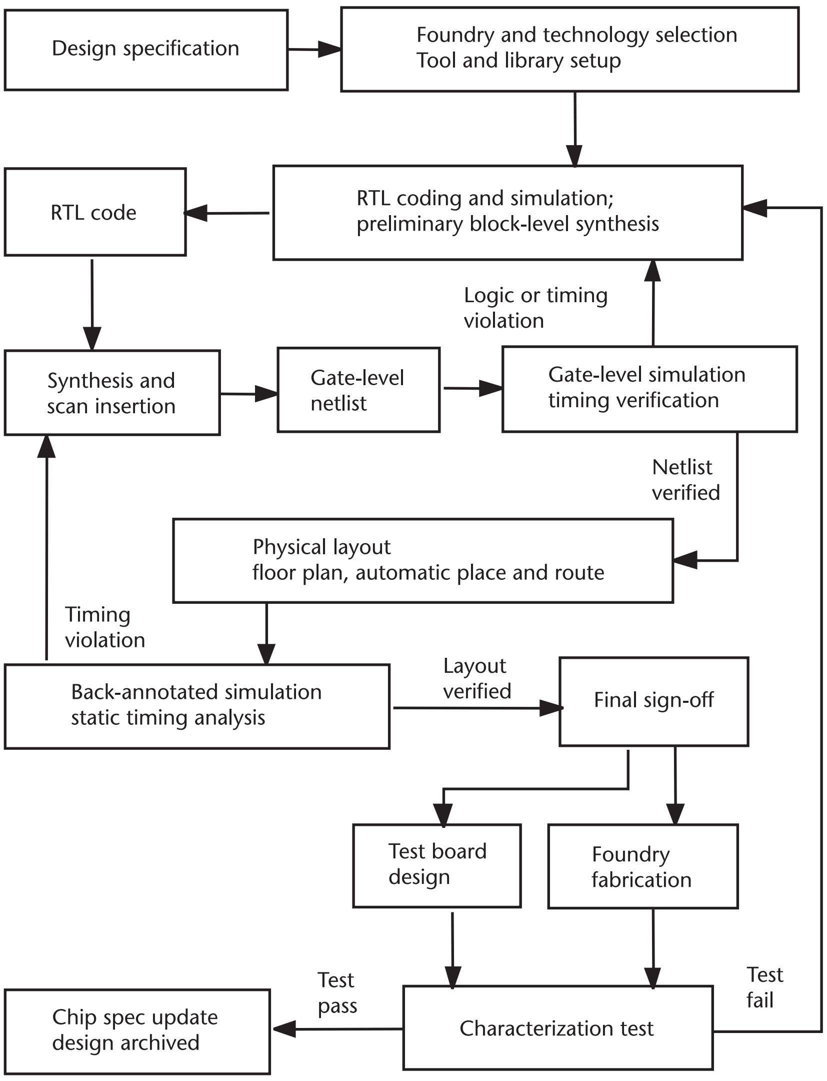


Figure 4.2 Digital IC design flow.

to guarantee the drive strength. Depending on the timing constraints, the synthesis tool will work to minimize the size or maximize the speed of the digital circuits.

The verified high-level RTL codes can be transferred into a gate-level netlist by an automated synthesis process. The gate-level netlist models the system using basic digital logic gates, such as NAND, NOR, and XOR gates and flip-flops. Those basic logic gates are designed at the transistor level by vendors. During the synthesis, an internal scan chain can be inserted to allow automated digital testing and fault detection. The gate-level code has to be resimulated to make sure that the synthesis has not altered the system's function. Next, a place-and-route tool can place the layout of basic logic gates and route them according to the gate-level code. In this automated layout process, the designer can also specify the desired floor plan and insert isolation rings to protect the sensitive blocks.

After layout, the wire delay information will be extracted and saved in a standard delay format (.sdf) file. This delay information will be back-annotated into the postlayout simulation, and the designers will then perform static timing analysis to make sure that the design meets the functional and timing specifications. Should the back-annotated simulation or static timing analysis fail, the design should be refined until all the specifications are met. Finally, the verified layout file in Graphic Data System II (GDSII) format is submitted to the semiconductor foundry for fabrication. Meanwhile, the designers need to design the test board and prepare for prototype testing and chip characterization. In the digital design flow, the two major design steps, synthesis and place/route, have been automated; thus, complicated functions can be implemented in digital designs. Nowadays, a complicated digital chip with multimillion gates can be implemented using the above-described digital design flow and achieve clock frequencies of a few hundred megahertz.

4.2 Verilog HDL

Verilog and VHDL are the two major HDLs widely used in digital IC designs in industry and academia. Both languages are now the Institute of Electrical and Electronics Engineers (IEEE) standard languages for digital hardware modeling and simulation. Verilog was developed by Philip Moorby [3] in 1983–1984 for Gateway Design System Corporation, a company later acquired by Cadence Design Systems. Verilog HDL was a proprietary language of Cadence until May 1990 with the formation of Open Verilog International. Cadence opened the language to the public domain with the hope that the market for Verilog-related software products would grow more rapidly with broader acceptance of the language. The IEEE standardized Verilog in 1995 as the IEEE 1364 standard. Verilog is very similar to the well-known C language and is thus easy to learn. VHDL was developed by the U.S. Department of Defense in 1983 and was made an IEEE standard in 1987 (IEEE 1067-1987). The standard was updated in 1993 and in 2001 as the IEEE 1076 standard. VHDL is very similar to Ada (a computer language developed by the military). Since few engineers have experience with Ada, VHDL may seem more challenging to learn than Verilog.

HDLs are used to describe digital systems at various levels. For example, an HDL might describe the layout of the wires, resistors, and transistors on an IC chip (i.e., at the switch level). Alternatively, it may describe the logical gates and flip-flops in a digital system (i.e., at the gate level). An even higher level describes the registers and the data transfers between the registers (i.e., at the RTL). Verilog supports all of these levels, allowing a hardware designer to describe designs at a high level of abstraction, such as at the architectural or behavioral level, as well as at the lower implementation levels, such as gate and switch levels. In the following sections, we will briefly introduce Verilog HDL, followed by examples of an MMD and a sigma-delta ($\Sigma\Delta$) modulator for fractional-N frequency synthesis. A detailed discussion of the Verilog language is beyond the scope of this book. Readers who are interested in advanced Verilog programming can refer to numerous references, such as [3–6].

4.2.1 Verilog Program Structure

Verilog's program structure resembles that of the C programming language. The Verilog HDL describes a digital block as a module. Verilog programs contain a set of modules in which the main module models the top level of a digital system. The main module instantiates the next lower-level modules, which further instantiate lower-level modules, forming a hierarchical program structure. A module in Verilog HDL takes inputs as parameters and returns output values, resembling a digital block with inputs and outputs. The interconnects among the modules are modeled in a higher-level module that instantiates those lower-level modules.

The structure of a module in Verilog HDL is specified as follows:

```
module <module name> (<port list>);
  Parameters (optional);
  Declarations <input>, <output>, <inout>, <wire>, <reg>;
  Instantiations <lower level modules>;
  Combinational or structural statements <assign>;
  Sequential or behavioral statements <always>, <initial>;
endmodule
```

A module always starts with a key word <module>. The <module name> is an identifier that uniquely names the module. The <port list> lists the input, input/output (inout), and output ports, which interconnect with other modules. The declarations of ports and optional <parameters> must come first, after the module definition. The <wire> declares the output signal of a combinational circuit. The <reg> specifies the output signal of a register (i.e., the sequential circuit), which holds its current value since a register is a memory device. Unlike a memory circuit, a combinational circuit does not hold its current value. The port declarations <input>, <output>, and <inout> can be specified as either <wire>, if it does not hold its current value, or <reg>, if it does hold its value. The <assign> statement can be used to model the output of a combinational circuit, while the <always> statement can be used to model the output of either a combinational or a sequential circuit. The <always> statement can be used for both behavioral and structural programming, while the <initial> statement can only be used for behavioral programming and is normally used in test benches. The procedural assignments have the form <reg variable> = <expression>, where the <reg variable> must be a register or memory. Procedural assignment may only appear in <initial> and <always> constructs. The use of those constructs can be understood through the following examples of a simple combinational circuit (a NAND gate).

Example 4.1: Behavioral Model of a Two-Input NAND Gate

```
module nand (a, b, c);
  input a, b;
  output c;
  wire c;

  assign c = ~ (a & b);

endmodule
```

In the example, the operator "~" means negation, and "&" denotes the logical AND operation. Verilog operators will be defined in the following section. The continuous assignment "assign" continuously monitors the changes to the right-hand-side variables and updates the left-hand-side variable whenever a right-hand-side variable is updated. The "assign" statement models combinational circuits where the outputs follow the input changes. The following example models a simple sequential circuit (a flip-flop).

Example 4.2: D-Flip-Flop Design with an Active-Low Reset (rst_n) and Clock-to-q Delay of 10 Time Units Using Behavioral Modeling

```
module d_ff(data, clock, rst_n, q, q_);
  input    data, clock, rst_n;
  output   q, q_;
  reg      q;
  always @ (posedge clock or negedge rst_n )

begin
  if (!rst_n) q <= 0; // If reset is low make output zero
  else q <= #10 data; // 10 is the clock to output q delay
end

assign q_ = ~q;

endmodule
```

When the synthesis tool synthesizes the above behavioral code, it will generate a gate-level structure for the positive-edge-triggered D-flip-flop with active-low reset, as illustrated in Figure 4.3. Note that the delay (#10) will be ignored by the synthesis tool. The delay used in RTL code is for simulation purposes only. The real delay for the flip-flop will be specified in the foundry's technology library. It is a useful practice to put all Verilog variables in lower case but figure labels in a combination of upper case, lower case, and subscripts to improve readability and

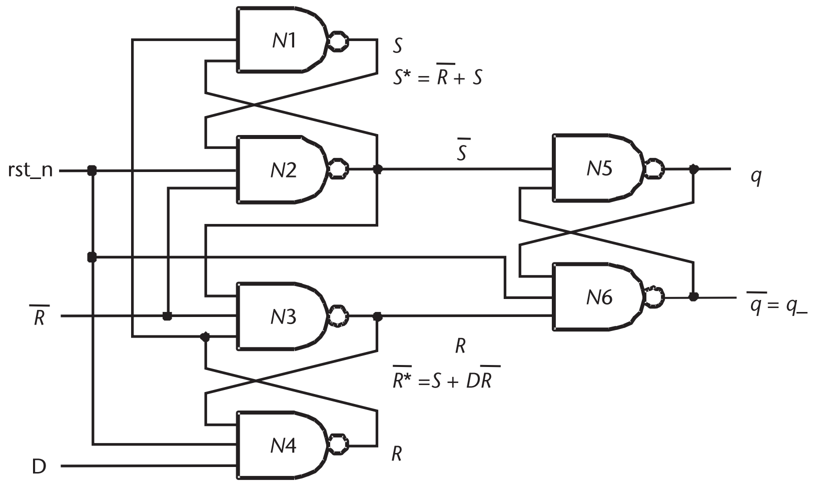


Figure 4.3 Synthesized gate-level structure for the positive-edge-triggered D-flip-flop with active-low reset.

appearance. However, to avoid confusion, we have here attempted to match the figure labels to the Verilog labels. As seen in Figure 4.3, the flip-flop uses six gates, and this information can be used to estimate the total number of gates in a digital design. More detail on gates at the transistor level will be given in Chapter 5.

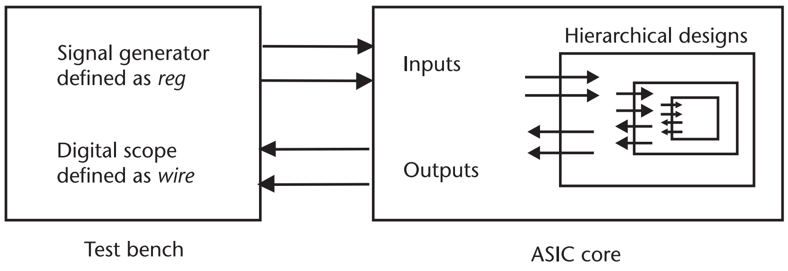
In the example, "reg" declares a 1-bit register. Each instantiation of this module will model a separate register. The "@" causes a wait-for-a-trigger event, in this case a positive (rising) clock edge or a negative reset. When one of these events occurs (i.e., at the clock rising edge), "always" continuously executes the statements between "begin" and "end," which evaluates "data," waits 10 time units, and then assigns q with the data value obtained 10 time units ago. These actions model a D-flip-flop with a clock-to-q delay of 10 time units. The time unit is normally specified in the test bench.

The module construct in Verilog is different from subroutines and functions in other procedural languages. Instead of being called, a module is instantiated at the start of the program and stays alive in the entire simulation, just as if we have soldered a circuit block and assume it is wired for the entire test. Verilog HDL is an event-driven program. Whenever the specified event occurs, it triggers the action to update the outputs. In the above NAND gate example, the output is updated whenever one of the inputs is updated. In the above flip-flop example, the output is updated whenever the clock rising edge occurs. Verilog is a concurrent language. It models concurrent activities and makes them look like they happened at the same time, which is very important for synchronous digital designs.

In Verilog, the general form to invoke an instance of a module is given by

```
<module name> <parameter list> <instance name> (<port list>);
```

where <parameter list> contains the values of parameters passed to the instance (see Example 4.3). Generally, each module is a separate file, although this is not required by the language syntax. A test bench can be written in a separate file, which instantiates the top-level module, applies the stimuli to its inputs, and evaluates its outputs. In the test bench, the stimuli generated in the test bench should be defined as "reg" since they are generated in the test bench and their values need to be stored, while all the signals under test should be defined as wire. This resembles the IC testing process as shown in Figure 4.4, where the signal generator generates the input signals for testing the application-specific integrated circuit (ASIC), and the outputs of the ASIC are captured by a digital scope for analysis. Only the top



Signal generator defined as *reg*
Digital scope defined as *wire*
Inputs
Outputs
Hierarchical designs
Test bench
ASIC core

Figure 4.4 Verilog modules and the test bench.

level of the ASIC core is interconnected with the test bench. Inside the ASIC core, a hierarchical design is used with lower-level modules being instantiated in higher-level modules.

Example 4.3 gives a complete Verilog model for a 4-bit ripple adder and a test bench that generates the adder inputs to test its performance. The 4-bit ripple adder instantiates a 1-bit full adder, which is built structurally based on

$$FA\begin{cases} sum = x \oplus y \oplus c\_in \\ c\_out = x \cdot y \cdot c\_in \end{cases} \tag{4.1}$$

where *c_in* is the carry-in bit, and *c_out* is the carry-out bit; *x* and *y* are two addends.

Example 4.3: A Verilog Model for a 4-Bit Ripple Adder

```
//Structural model for a 1-bit full adder:
module fulladder(sum, c_out, x, y, c_in);
  output sum, c_out;
  input x, y, c_in;

assign sum = x ^ y ^ c_in;    // ^ denotes XOR
assign c_out = x & y & c_in;  // & denotes AND, | for OR

endmodule

//Structural model for a 4-bit ripple adder:
module FourBitAdder(sum, c_out, x, y, c_in);
  output [3:0] sum; //[3:0] specifies the no. of bits for sum.
  output c_out;
  input [3:0] x, y;
  input c_in;
  wire c1, c2, c3; //represent the interconnections between adders.

//Instantiation of 4 1-bit full adders:
  fulladder fa0(sum[0], c1, x[0], y[0], c_in);
  fulladder fa1(sum[1], c2, x[1], y[1], c1);
  fulladder fa2(sum[2], c3, x[2], y[2], c2);
  fulladder fa3(sum[3], c_out, x[3], y[3], c3);

endmodule
```

A test bench for the 4-bit ripple adder:

```
‘timescale 1ps / 1ps // timescale: ref_time_unit/precision
module test(); //no inputs and outputs
  reg c_in; //define stimulus generated in testbench as reg
  reg [3:0] x, y;
  wire c_out; //define signals under test as wire
  wire [3:0] sum;

  FourBitAdder A1(sum, c_out, x, y, c_in);

initial
  $monitor($time,,,"x=%d y=%d ci=%d s=%d co=%d", x,y,c_in,sum,c_out);
```

```
initial
  begin //generates x
     x = 4'b0001;
    #25000 x = 4'b0001;
    #25000 x = 4'b0010;
    #25000 x = 4'b0001;
    #25000 x = 4'b0001;
    #25000 x = 4'b1000;
    #25000;
  end

initial
  begin //generates y
     y = 4'b0001;
    #25000 y = 4'b0010;
    #25000 y = 4'b0011;
    #25000 y = 4'b1111;
    #25000 y = 4'b1111;
    #25000 y = 4'b1111;
    #25000;
  end

initial
  begin //generates c_in
     c_in = 1'b0;
    #100000 c_in = 1'b1;
    #25000 c_in = 1'b0;
    #25000;
  end

initial
  #150000 $finish;

endmodule
```

Figure 4.5 shows the input and output waveforms when the adder module and the test bench are simulated in a Verilog simulator.

The monitor statement in the example prints the following outputs:

```
0 x = 1 y = 1 ci = 0 s = 2 co = 0
25000 x = 1 y = 2 ci = 0 s = 3 co = 0
50000 x = 2 y = 3 ci = 0 s = 5 co = 0
75000 x = 1 y = 15 ci = 0 s = 0 co = 1
100000 x = 1 y = 15 ci = 1 s = 1 co = 1
125000 x = 8 y = 15 ci = 0 s = 7 co = 1
```

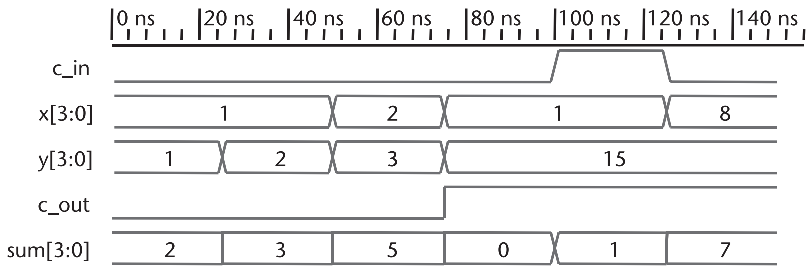


Figure 4.5 Simulated input and output waveforms for the 4-bit ripple adder.

Note that anything after "//" is a comment and will be ignored by the compiler. The ‘timescale 1ps/1ps denotes that the reference time unit is 1 ps, and the simulation precision is 1 ps. In the test bench, the statements in the block of the "initial" constructs will be executed sequentially, generating data inputs for the adder. The "$monitor" prints its string when one of the listed variables changes. The extra commas after $time,,, print extra spaces. The #150000 $finish stops the program after 150000 ps = 150 μs.

For display, the commonly used formats are

`%b`	Display in binary format;
`%c`	Display in ASCII character format;
`%d`	Display in decimal format;
`%h`	Display in hex format;
`%o`	Display in octal format;
`%s`	Display in string format.

Escape sequences may be included in a string. The commonly used escape sequences are the following:

`\n`	The newline character;
`\t`	The tab character;
`\\`	The "\" character;
`\"`	The """ character.

4.2.2 Verilog Data Formats

In Verilog HDL, the reg and wire data objects may have the following four logic values:

`0`	Logical zero or false;
`1`	Logical one or true;
`x`	Unknown logical value;
`z`	High impedance of tristate gate.

The reg and wire variables are initialized to x at the start of the simulation. Any unconnected wire variable has the x value. x is used as a debugging aid since it shows that the simulator cannot determine the answer. A tristate gate drives its output to be either a zero, a one, or a high impedance (z). The size of a register or wire can be specified in the declaration as

```
reg [15:0] a;
wire [0:7] b;
```

where the bit notation is [<start-bit>:<end-bit>]. Thus, a is a 16-bit register with the fifteenth bit as the most significant bit (MSB), and b is an 8-bit wire with the zeroth bit as the MSB. The above declarations can also be parameterized using the parameter constructs:

```
parameter size = 16;
reg [size-1:0] a;
```

or the define constructs:

```
‘define size 8;
reg [0:size-1] b;
```

In Verilog, the data can be represented in binary format with notation b, decimal format with d, octal format with o, and hex format with h. A 3-bit binary word 000 can be represented as 3’b000. Concatenation is a very useful format for writing data. The {a, b} construct means the bits of *a* and *b* are concatenated together. For example, if size = 8, C = {1’b1, {(size-2){1’b0}}, 1’b1}) = 8’b10000001, where {(size-2){1’b0}} represents (size-2) zeros and {1’b0} and {1’b1} represent 1-bit words with values of zero and one, respectively.

For convenience in behavioral modeling, Verilog HDL has several data types, such as integer, real, and time, that do not represent hardware implementations. Data types integer and real have a similar definition as those in C languages, while time variables hold 64-bit quantities and are used in conjunction with the $time system function.

4.2.3 Verilog Operators

Table 4.1 summarizes some common Verilog operators. Their applications will be become clear through Table 4.2. The table illustrates the usage of some Verilog operators that may be confusing to beginners.

4.2.4 Verilog Control Constructs

Verilog HDL has a rich collection of control constructs that is very similar to a procedural language such as the C language. Instead of C’s { } parentheses, Verilog uses begin and end. The meanings of the if and else statements are the same as those defined in the C language. For instance,

```
if (a = = 2’b00) b = 2’b11; //one line can be combined with if statement.
else
  begin //more lines can be inserted between begin and end.
    b = 2’b10;
    c = 2’b01;
  end
```

The for statement is very close to what is defined in C except that the “++” and “--” operators do not exist in Verilog. Therefore, we need to use i = i + 1.

```
for(i = 0; i < 10; i = i + 1) begin
  <statements>
end
```

Verilog is a discrete, event-driven simulator, and the events are scheduled for discrete times. During the processing, more events may be created and placed in

Table 4.1 Common Verilog Operators

Binary Arithmetic Operators		*Logical Operators*	
+	Addition	!	Logical negation
–	Subtraction	&&	Logical AND
*	Multiplication	||	Logical OR
/	Division	==	Logical equality
%	Modulus	!=	Logical inequality
Bitwise Operators		*Unary Operators*	
~	Bitwise negation	–	Unary minus
&	Bitwise AND	&	Unary reduction AND
|	Bitwise OR	|	Unary reduction OR
^	Bitwise XOR	^	Unary reduction XOR
~&	Bitwise NAND	~&	Unary reduction NAND
~|	Bitwise NOR	~|	Unary reduction NOR
~^ or ^~	Bitwise XNOR	~^ or ^~	Unary reduction XNOR
Relational Operators		*Other Operators*	
>	Greater than	{,}	Concatenation
>=	Greater than or equal to	<<	Shift left
<	Less than	>>	Shift right
<=	Less than or equal to	?:	Conditional assignment
===	Equality (bitwise comparison)		
!==	Inequality (bitwise comparison)		

Operator Precedence

Operators on the same line have the same precedence and associate left to right in an expression. Parentheses can be used to change the precedence.

Unary operators: ! & ~& | ~| ^ ~^ + – (highest precedence)
* / %
+ –
<< >>
< <= > >=
== != === !==
& ~& ^ ~^
| ~|
&&
||
?: (lowest precedence)

Table 4.2 Use of Verilog Operators

Operators	*Definitions*	*Examples*
!	Logic negation.	!(2'b01 == 2'b10) returns 1
==	Logic equality. If any of the bits is unknown, the result is unknown.	(2'b01 == 2'b10) returns 0
===	Case equality with bitwise comparison including comparison of *x* and *z* values. All bits must match for equality.	(4'b0xz1 === 4'b0xz1) returns 1 (4'b0xz1 === 4'b0x01) returns 0
~	Bitwise negation.	~4'b0110 = 4'b1001;
&	Bitwise AND.	4'b0110 & 4'b1110 = 4'b0110;

&, |, ^	Unary AND/OR/XOR, producing a single bit AND/OR/XOR of all the bits.	&(4'b0110) = 0; |(4'b0110) = 1;
		^(4'b0110) = 0; ^(4'b1110) = 1;

<<	Shift left and fill the vacated bit positions with zeros.	8'b01100011<<2 = 8'b10001100 shifts 2 bits to the left with zero filled at the right-most bits. Equivalent to multiplying by 4.
?:	Conditional assignment. A = B > C ? D : E; if B > C is true, A will be assigned D, otherwise A will be assigned E.	c = 4'd5 > 4'd6 ? 1 : 0; c will be assigned 0 since 5 > 6 is false.

the queue. When all the events of the current time have been processed, the simulator advances time and processes the next events in the queue. If there is no timing control, the simulation time does not advance. Simulated time can progress only by one of the timing controls.

Verilog HDL provides three types of timing control for event-driven simulations. The first type is the delay control ("#"), which specifies the time duration between the initial appearance of the statement and its actual execution. For example, #10 a = b & c; provides a delay of 10 time units before executing the procedural assignment statement.

The second type is the event expression always @, which allows statement execution when the event happens. The execution of a procedural statement can be triggered by edges of variable transitions or value changes on a wire or register. For example, @ sel triggers an event when sel changes value, and @(posedge clock) is controlled by the positive edge of the clock. For posedge and negedge, the following variable must be a 1-bit expression, typically a clock.

The third type involves the wait statement, which waits for a specific variable to change. The construct wait is used to wait for an expression to become TRUE because a variable in the expression is changed by another process. For example, wait (a == 1) will wait for variable a to become 1. Construct wait is level sensitive compared to the edge-triggered statement @(posedge clock).

If the expression for a while statement is TRUE, the simulator will continuously execute the loop. Another process cannot stop the while loop once it is running. Thus, while cannot be used to wait for a change in an input to the process, and there must be a delay operator such as "#" or "@" inside the while loop to stop the process. The following example creates an endless loop.

Example 4.4: An Endless While Loop

```
module EndlessWhileLoop (a);
  input a;
  always
    while(a) $display ("This is an endless loop");
endmodule
```

The repeat statement repeats the statements between begin and end for *n* times:

```
repeat (n) begin
  <statements>
end
```

4.2.5 Blocking and Nonblocking Assignments

Verilog HDL has two types of procedural assignment statements: blocking and nonblocking assignments. The blocking assignment statement ("=" operator) acts the same as the C language statement. The whole assignment statement is executed before control passes on to the next statement. For blocking assignment a = b, b is calculated and used immediately to update a. The next statement that uses a as a source will use the updated value of a. The blocking assignment is normally used to assign combinational outputs.

In contrast, the nonblocking assignment statement ("<=" operator) evaluates all of the right-hand sides in a program for the current time and assigns the left-hand sides at the end of the time unit simultaneously. For nonblocking assignment a <= b, b is calculated, and a nonblocking update event is scheduled for a at the current time. Execution of the following statements continues. The new value of a will not be seen by other elements until the nonblocking update events are executed. The nonblocking assignment was specially developed for Verilog HDL to model synchronous digital designs. It synchronizes the assignments so that multiple nonblocking assignments can be executed concurrently. The nonblocking assignment is normally used to assign register outputs in synchronous sequential circuit designs.

The subtle difference between the blocking and nonblocking assignments can be understood through Example 4.5 and the discussion following the example. The simulation results for this example are shown in Figure 4.6.

Example 4.5: Blocking and Nonblocking Assignments

```
module test();
  reg[4:0] a, e1, e2, e3, e4, e5;
  reg[4:0] g1, g2, g3;

  initial begin
    a=0; e1=0; e2=0; e3=0; e4=0; e5=0; g1=0; g2=0; g3=0;
    #100 $finish;
  end
  initial forever
  #10 a <= a+1;

//Nonblocking Assignments, Shift Register
  always @(a) begin
    e1 <= a;
    e2 <= e1; //e2 is updated with old value of e1
    e3 <= e2;
  end
```

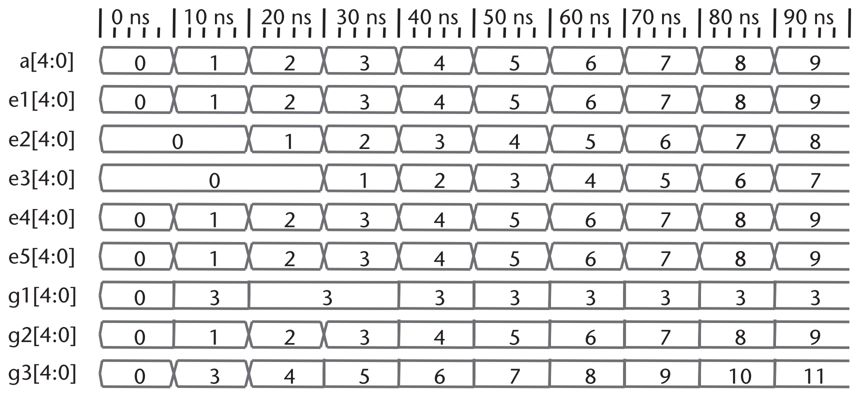


Figure 4.6 Simulation diagram for comparison between the blocking and nonblocking assignments.

```
//Blocking Assignments, Wire
  always @(a) begin
    e4 = a;
    e5 = e4; //e5 is updated with new value of e4
  end

always @(a) begin
g1 = a; //line 28
g2 = 2; //line 29
g3 = 3; //line 30
g1 <= g3; //line 31
g2 <= g1; //line 32
g3 = g1+g2; //line 33
end

endmodule
```

In the example, e1, e2, and e3 are assigned with nonblocking assignments. Therefore, they all get the old values of the right-side variables at the beginning of the current time, and they are updated with the new values at the end of the current time (beginning of the next event). This reflects how register transfers occur in real hardware. Registers e1, e2, and e3 actually form a 3-bit shift register with the input data a. Registers e4 and e5 are assigned with blocking assignments. Therefore, they all get the new values of the right-side variables at the beginning of the current time. As a result, e4 and e5 all follow variable a without delay. Registers g1, g2, and g3 are assigned with a mixture of blocking and nonblocking assignments. From line 28 to line 30, they are assigned with the blocking assignments. Therefore, the statements after line 30 should get the assigned new values. Since g1 gets g3's new value of 3 at line 31 and g2 gets g1's new value a at line 32, g1 is a constant 3, and g2 follows variable a as shown in Figure 4.6. Note that lines 31 and 32 are nonblocking assignments. Hence, g1 and g2 in line 33 should take the old values assigned before line 31 and line 32. Thus, g3 is assigned with a + 2 on line 33.

4.2.6 Tasks and Functions

Verilog HDL provides task and function constructs to allow the behavioral description to be partitioned into manageable parts. The subtle differences between task and function can be summarized as follows:

1. *Inputs and outputs:* Tasks may have zero or more arguments and do not return a value to an expression. However, values written by task can be copied back through its input and output ports. A function has at least one input and returns a single value to the expression that called it.
2. *Timing controls:* Functions must execute during one simulation time unit. That is, no time-controlling statements, such as delay control ("#"), event control, ("@") or wait statements, are allowed in a function. In contrast, a task may contain time-controlled statements.
3. *Invocation:* A task call is a separate procedural statement. A function call is an operand in an expression. A Verilog function can invoke (call, enable) another function but not a task, whereas a task may call other tasks and functions.

The definition of a task is given as follows:

```
task <task name>; //no parameter list or ( )
  <argument ports>
  <declarations>
  <statements>
endtask
```

An invocation of a task takes the following form: <name of task> (<port list>), where <port list> is a list of expressions that correspond by position to the <argument ports> of the definition. Example 4.6 demonstrates how to use the task construct.

Example 4.6: Using the Task Construct

```
module tasks;
  reg c;

task and_gate; // task definition
  input a, b; // input argument ports
  output c; // output argument port
  assign c = a & b;
endtask

initial begin
  and_gate(0, 1, c); // invocation of task and_gate
  $display ("c= %b", c);
end

endmodule
```

The purpose of a function is to return a value that is to be used in an expression. The definition of a function is given as follows:

```
function <range or type> <function name>; // no parameter list or ( )
  <argument ports>
  <declarations>
  <statements>
endfunction
```

where <range or type> is the type of the results passed back to the expression. Example 4.7 is a function that calculates the carry-out and sum bits of a 1-bit adder.

Example 4.7: Using the Function Construct

```
module functions;
reg [1:0] d;

function [1:0] adder; // function definition
  input a, b; // two input argument ports
  assign adder = { a|b, a^b }; //concatenate carry-out and sum bits
endfunction

initial begin
  d = adder(1,1); // invocation of function adder
  $display ("d = %b", d);
end

endmodule
```

4.3 Behavioral and Structural Modeling

Digital logic modules can be specified behaviorally or structurally or with a combination of the two. A behavioral module defines the behavior of a digital system using abstracted programming language constructs (e.g., if and for statements). A structural specification expresses the behavior of a digital system as a hierarchical interconnection of submodules. In structural modeling, the system is modeled based on its structural schematics. The gate-level models are structural models, where the system is represented using primitive gates and their interconnections. At the bottom of the hierarchy, the components must be primitives or specified behaviorally. Verilog HDL has built-in gate-level primitives such as NAND, NOR, AND, OR, XOR, XNOR, BUF, NOT, and so forth. Using gate-level primitives, Example 4.8 gives the structural model for a 2:1 multiplexer. The structural code is nothing but a description of the multiplexer hardware schematics shown in Figure 4.7. For the same circuit, we write a behavioral model using if and else as shown in Example 4.9. The behavioral model directly specifies the truth table of the 2:1 multiplexer; that is, if sel=1, c=a, and if sel=0, c=b. However, there are subtle things that should be considered when a combinational circuit is modeled using a procedural behavioral model, as is explained in the next section.

Example 4.8: Structural Module of a 2:1 Multiplexer

```
module mux (c, sel, a, b);
  output c;
  input sel, a, b;
  wire d, e, sel_;
  and g1(e, a, sel),
      g2(d, b, sel_);
  or g3(c, d, e);
  not g4(sel_, sel);
endmodule
```

Example 4.9: Behavioral Module of a 2:1 Multiplexer

```
module mux (c, sel, a, b);
  output c;
  input sel, a, b; //input set
  reg c; //combinational output
  always @ (sel or a or b) //sensitivity list
    if (sel == 1) c=a; //control path
    else c=b; //control path
endmodule
```

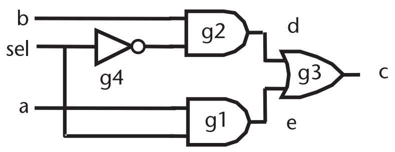


Figure 4.7 Structure of a 2:1 multiplexer.

4.4 Combinational Digital Circuit Design

Combinational circuits can be modeled structurally using continuous assignments with the keyword assign. Continuous assignments drive wire variables and are evaluated and updated whenever an input operand changes value. The left-hand variable of an assign statement should be defined as wire. For instance, <assign c = a & b;> ANDs the values on wire a and b and passes the value to wire c continuously. Combinational circuits can also be modeled using a procedural assignment such as the always construct. However, caution needs to be taken to prevent mistakenly converting a combinational circuit into a sequential circuit. Let us evaluate a behavioral model for a combinational circuit in Example 4.9, where we intend to model a 2:1 multiplexer. Note that the output of the multiplexer c has to be defined as a reg in order to be able to assign it inside the always block. In Example 4.9, the sensitivity list of the always construct (the variables in the brackets following the @) observes any changes to a, b, or sel, then executes the block containing the if and else statements. However, the multiplexer does not need a register. The register here is just an "artifact" of the descriptions. Inside the always block, registers are found on the left-hand side of the assignment statements. How can the synthesis tool figure out that this is a combinational circuit? Recall the prerequisites of a combinational circuit: (1) the output is only a function of the *current* inputs, and (2) anytime an input changes, the output will be updated. Hence, if the following rules are satisfied, combinational logic can be modeled using the procedural statements:

1. Every element of the input set must be in the sensitivity list.
2. The combinational output must be assigned in every possible control path.

In other words, any input changes (they are all listed in the sensitivity list) will trigger an update of the output (it has been assigned in every possible control path). Now, the synthesis tool can interpret the code in Example 4.9 as combinational logic.

In Verilog HDL, a case statement is widely used in modeling multiplexers and conditional branches. The following example models a 4:1 multiplexer, where a and b are control bits. For the branches of ({a, b}) that have not been covered (e.g., if {a, b} = 2'b11 in the example), the output will take the value specified in the default statement. For synthesis purposes, it is good practice to have a default statement in each case statement, even if all possible branches have been listed. This will prevent mistakenly modeling a combinational circuit as a sequential circuit.

Example 4.10: A 4:1 Multiplexer Using a Case Statement

```
module multiplexer (f, a, b);
  output f;
  input a, b;
  reg f;

  always @ (a or b)
    case ({a, b})
    2'b00: f=1'b1;
    2'b01: f=1'b0;
    2'b10: f=1'b0;
    default: f=1'b0; // used if {a, b}=2'b11.
  endcase

endmodule
```

The case statement can also be used to model any combinational circuit if its truth table is given. For the truth table shown in Figure 4.8, we can simply represent its entries using case statements as shown in Example 4.11, where the default output is don't care x. If a default statement is included, the synthesis tool will correctly synthesize the code as a combinational circuit since all the possible control paths have been covered.

Example 4.11: Combinational Circuit Design Using a Case Statement with the Truth Table Given

```
module combinational(f, x, y, z);
output f;
input x, y, z;
reg f;

always @ (x or y or z) begin
  case ({x,y,z})
    3'b010 : f=1'b0;
    3'b011 : f=1'b1;
    3'b100 : f=1'b1;
    3'b110 : f=1'b0;
    3'b111 : f=1'b1;
    default: f=1'bx;
  endcase
end

endmodule
```

4.5 Sequential Digital Circuit Design

In order to synthesize the RTL code successfully, designers should follow the rules for sequential logic design carefully:

1. Use the always block. The sensitivity list of the always block should only list the inputs that may cause state changes, such as the clock edge and reset.

z \ xy	00	01	11	10
0	x	0	0	1
1	x	1	1	x

Figure 4.8 Truth table for a combinational circuit.

2. Do not specify the clock condition inside the always block.
3. Specify the reset condition first. For instance, if(~reset) will start an active-low reset statement.
4. Use nonblocking assignments ("<=") to specify the sequential logic outputs. The nonblocking assignments inside the always block assume all data transfers occur at the clock edge concurrently, which forms the basis of synchronous digital designs.

Example 4.2 illustrates a simple flip-flop model that follows the above rule for sequential logic designs. In traditional sequential circuit design, designers start by building state tables or state diagrams for their system. Then, they need to go through state minimization, state assignment, and the excitation maps of the chosen flip-flops to find the logic implementation of their systems. Owing to logic optimization in digital synthesis, the designers now do not need to optimize their sequential logic circuits manually. Almost everything can be modeled behaviorally, and the synthesis tool will take care of the logic implementation automatically. As an example of sequential circuit design, a programmable frequency divider is modeled next, which can be used in PLL frequency synthesizer designs.

Example 4.12: Frequency-Divider Design

Design a frequency divider with division ratio programmable from 1 to 8. The output pulse should have close to a 50% duty cycle.

Solution: A frequency divider is basically a counter, which counts the input pulses and generates an output pulse every div input pulses, where div is the divider ratio. Output pulses with a 50% duty cycle can be generated if we allow the output pulse transition to occur in the middle of div input pulses or, in other words, a transition occurs at div/2 input pulses. If div is an even number, the output duty cycle will be exactly 50%. If div is an odd number, the output duty cycle will be close to 50%. The following program uses a parameter declaration to define the bit length. In this way, the program is parameterized for modeling frequency dividers with any division ratio. In this example, we choose size = 3 to program the ratio from 1 to 8. Note, the parameterized style of concatenating bits (e.g., {{(size-1){1'b0}},1'b1}) actually gives a 3-bit word of 3'b001.

```
module prodiv (fi, div, rst_, fo); // Programmable frequency divider

parameter size = 3; //divider counter bit length, division ratio from 1 to 8
input     fi; //input signal with frequency fi to be divided
input[size-1:0] div; //division ratio from 1 to 7, div=0 corresponds to divide by 8
input     rst_; //active-low reset
output     fo; //output signal with frequency fo = fi/div
reg[size-1:0] p; //p=1+integer(div/2) determines the falling edge of fo
reg[size-1:0] counter; //counter value;
reg      p0, fout; //output buffer
wire     ctr;

// Down counter: counter = counter - 1:
always @(posedge fi or negedge rst_)
  if (!rst_) //reset to div = 1
  begin
    counter[size-1:0] <= {{(size-1){1'b0}},1'b1};
    //The above line can be simplified as counter <= {{(size-1){1'b0}},1'b1}.
    //But, it's a good practice to specify clearly the every word range.
```

```
      p[size-1:0] <= {{(size-1){1'b0}},1'b1};
      p0 <= 1'b1;
    end
    else if (counter[size-1:0] = = {{(size-1){1'b0}},1'b1})
    begin //load new division ratio when counter=1
      counter[size-1:0] <= div[size-1:0]; //load new division ratio
      p0 <= (div[size-1:0] = = {{(size-1){1'b0}},1'b1}); //p0=1, if div=1
      if(div[size-1:0] = = {size{1'b0}})
        p <= {1'b1,{(size-2){1'b0}},{1'b1}}; //div=0 is used to divide by 8
      else
        p <= {1'b0,div[size-1:1]} + {{(size-1){1'b0}},1'b1}; //p=1+integer(div/2)
    end
    else //otherwise counter-1
      counter[size-1:0] <= counter[size-1:0] - {{(size-1){1'b0}},1'b1};
  // Output pulse generator:
  always @(posedge fi or negedge rst_)
    if (!rst_) fout <= 1'b0;
    else if(counter[size-1:0] = = {{(size-1){1'b0}},1'b1}) fout <= 1'b1;
    else if(counter[size-1:0] = = p) fout <= 1'b0;
  assign fo = p0 ? fi : fout; //if div=1, pass fi to fo

  endmodule //close prodiv
```

We can write a simple test bench to test the above code. We used another construct, ʻdefine, for parameterized programming, where the accent sign " ʻ " is required to stay with the parameter. Another way to instantiate a module in Verilog is also given in the following test bench, where variable names in the original module port list and the name used in the current module can be clearly seen. Also note that we use the "initial forever" construct to generate a clock signal that can run forever until the time defined by finish. The simulated divider input and output waveforms are shown in Figure 4.9, which clearly demonstrates the divide-by-two and divide-by-five waveforms.

```
'timescale 1ns / 1ns
'define Tclk 5 //clock period = 'Tclk*2 = 10ns;
'define size 3 //divider bit length;

module test();    // Test bench for programmable frequency divider
reg fi, rst_;     //define signal generated in test bench as reg
reg['size-1:0] R; //division ratio programmable from 1 to 7
wire fo;          //define signal captured by test bench as wire

//Instantiate prodiv module:
prodiv divider( .fi(fi),
//.<name in original port list>(<name in current module>)
                .div(R),
                .rst_(rst_),
                .fo(fo) );
```

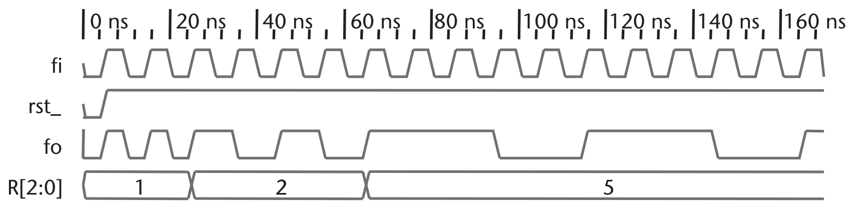


Figure 4.9 Simulated waveform for programmable frequency divider.

```
//Input signal generation:
initial begin
  fi <= 1’b0;
  rst_<= 1’b0;
  R <= {{(‘size-1){1’b0}},1’b1};
//Run test:
  #5     rst_ < = 1’b1; //release reset
  #(‘Tclk*2*2) R <= 2;  //After 2 clock cycles, load division ratio R = 2
  #(‘Tclk*2*4) R <= 5;  //After 4 clock cycles, load division ratio R = 5
  #(‘Tclk*2*11) $finish;    //Finish after 11 clock cycles
end
//clock generation:
initial forever #‘Tclk fi <= !fi;  //clock period = ‘Tclk*2

endmodule //close test
```

4.6 Digital Design Example I: A Multimodulus Divider

Although the frequency divider just discussed is programmable, its division ratio can be programmed only statically; therefore, the division ratio cannot be toggled at the input frequency. For frequency synthesis, we often need multimodulus dividers (MMDs) with modulus control that can be toggled faster. In this section, we will present such an MMD design as one of the examples for digital design using Verilog HDL. The architecture of the MMD is based on [7], which will be studied in Chapter 6 in detail. The MMD consists of three cascaded, 2/3, dual-modulus prescalers as shown in Figure 4.10.

The divide-by-2/3 cells can be implemented in various topologies, such as the one shown in Figure 4.11. The last cell in the MMD generates one mod pulse once in a division period. This signal then propagates “up” the chain and is reclocked by each cell along the way. An active-high mod signal enables a 2/3 cell to divide by three once in a division cycle if its division-programming input R is set high. If either R or mod is set to zero, the cell is set to divide-by-two mode. For a 3-bit MMD using three cascaded 2/3 cells, the programmable division ratio can be found by

$$N_{\mathrm{MMD}} = 8 + 4R_2 + 2R_1 + R_0 \tag{4.2}$$

Thus, a 3-bit MMD provides a programmable division ratio from 8 to 15 with step size of 1.

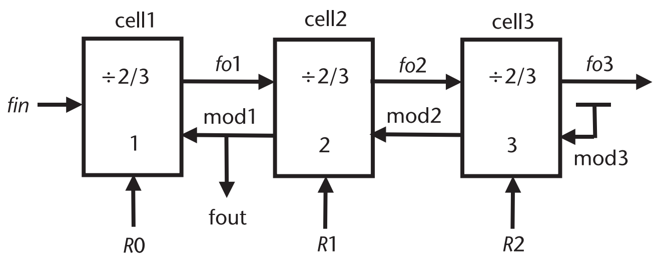


Figure 4.10 MMD using cascaded divide-by-2/3 cells.

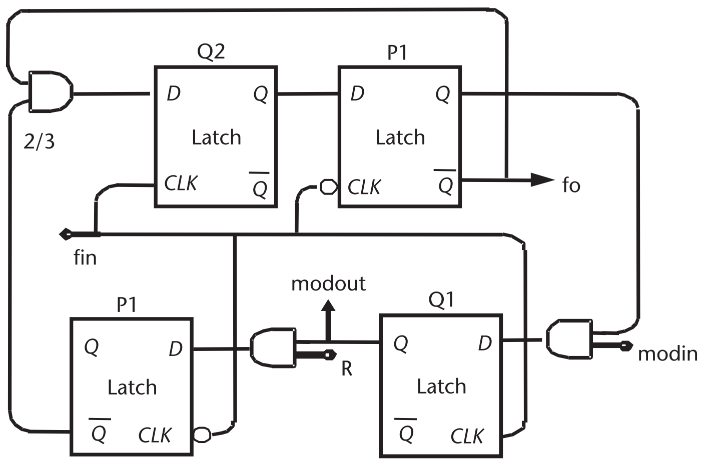


Figure 4.11 A dual-modulus divide-by-2/3 cell used in MMD design.

To model the MMD using Verilog HDL, we start with the latch module. Notice that the latch is a level-triggered device. Compared to an edge-triggered flip-flop, we use <always @(clk or rst_ or D)> without edge specifier posedge or negedge to model the latch behaviorally as follows:

```
module latch(Q, QBar, D, clk, rst_); // level-triggered latch behavioral model
  input D, clk, rst_;
  output Q, QBar;
  reg Q;
  wire QBar;

  always @(clk or rst_ or D)
  begin
    if(!rst_) Q<=1'b0;
    else if(clk)Q<=D;
  end
  assign QBar = ~Q;

endmodule
```

Next, we model the divide-by-2/3 cell, which can be done structurally by wiring the latches and AND gates as sketched in Figure 4.11.

```
//structural model of 2/3 cell based on Figure 4.11:
module DivBy2_3cell(modout, fo, modin, fin, R, rst_);
output modout, fo;
  input fin, modin, R, rst_;
  wire a, b, c, c1, d, e, f, f1, g, h, h1, i, i1;

  and (b, i, modin);
  and (e, modout, R);
  and (g, f1, fo);
  latch Q1 (modout, c1, b, fin, rst_),
    P1 (f, f1, e, ~fin, rst_),
    Q2 (h, h1, g, fin, rst_),
    P2 (i, fo, h, ~fin, rst_);

endmodule
```

Note that the reset signal may not be needed in hardware implementation if the initial output of the latch is not a concern. However, a reset signal is needed in the Verilog code for simulation purposes. Otherwise, the divider output will always be the unknown value *x* since feedback is involved in the divide by 2/3 cell. With the 2/3 cell module, we can model the 3-bit MMD using the cascaded topology shown in Figure 4.10:

```
//structural model of a 3-bit MMD based on in Figure 4.10:
module MMD(fout, fo3, fin, R0 ,R1, R2, rst_, mod3);

  output fout, fo3;
  input R0, R1, R2, mod3, fin, rst_;
  wire mod0, mod2, fo1, fo2;

  DivBy2_3cell cell1(mod0, fo1, fout, fin, R0, rst_),
    cell2(fout, fo2, mod2, fo1, R1, rst_),
    cell3(mod2, fo3, mod3, fo2, R2, rst_);

endmodule
```

Finally, let us design a test bench to test the MMD performance. Although divider performance can be evaluated by observing its input and output waveforms, it is highly desirable to design a test bench with a self-checking capability, especially when the design is highly complex. For the MMD, we design a test bench that can automatically count the input and output pulses and, thus, tells the designer what the simulated division ratio is.

```
‘timescale 1ns / 1ns
‘define Tclk 5

module test();
  reg fin, rst_;
  reg[2:0] R;
  wire fout, fo3;
  integer count_r, count_r1, count_r2, N;

  MMD mmd( .fout(fout),
      .fo3(fo3),
      .fin(fin),
      .R0(R[0]),
      .R1(R[1]),
      .R2(R[2]),
      .rst_(rst_),
      .mod3(1’b1) ); // mod3 in the last cell is connected to logic 1.
  initial begin
    fin <= 1’b0;
    rst_ <= 1’b0;
    #5 rst_=1’b1;
    R = 3’b000;
    $display("Performing self checking for R=000, MMD_ratio=8");
    #85 R = 3’b100;
    #80 $display("Performing self checking for R=100, MMD_ratio=12");
    #160 $finish;
  end

  initial forever
    #‘Tclk fin <= ~fin;

// Self checking:
  always @(negedge fin) //count_r counts input pulses
```

```
        begin
         if(~rst_) count_r=0;
         else count_r=count_r+1;
        end
      always @(negedge fout) //count_r counts output pulses
        begin
         count_r2=count_r1;
         count_r1=count_r;
         N = count_r1-count_r2; //N counts no. of input pulses within 1 output cycle
         $display("count_r1=%0d, count_r2=%0d, N=%0d", count_r1, count_r2, N);
        end

    endmodule
```

The simulated MMD waveforms are shown in Figure 4.12. The waveforms demonstrate that the MMD divides the input frequency by eight when the modulus inputs R = 3’b000 and by 12 when the modulus inputs R = 3’b100. Note that the fout and fo3 signals have the same frequency, but with different duty cycles. The self-checking test bench also prints the following messages, indicating that the MMD is dividing by 8 and 10 after loading the two modulus inputs:

```
count_r1=0, count_r2=0, N=0
Performing self checking for R=000, MMD_ratio=8
count_r1=8, count_r2=0, N=8
count_r1=16, count_r2=8, N=8
Performing self checking for R=100, MMD_ratio=12
count_r1=28, count_r2=16, N=12
```

4.7 Digital Design Example II: A Programmable MASH ΣΔ Modulator

To demonstrate digital logic design for frequency synthesis, we discuss the Verilog model for a third-order multiloop ΣΔ modulator discussed in [8]. The ΣΔ modulator as illustrated in Figure 4.13 will be discussed in Chapter 9. The ΣΔ modulator consists of three cascaded accumulators with a single-bit carry from each accumulator. The carries from three accumulators are passed on to z^{-1} delay blocks and are then used to select coefficient banks.

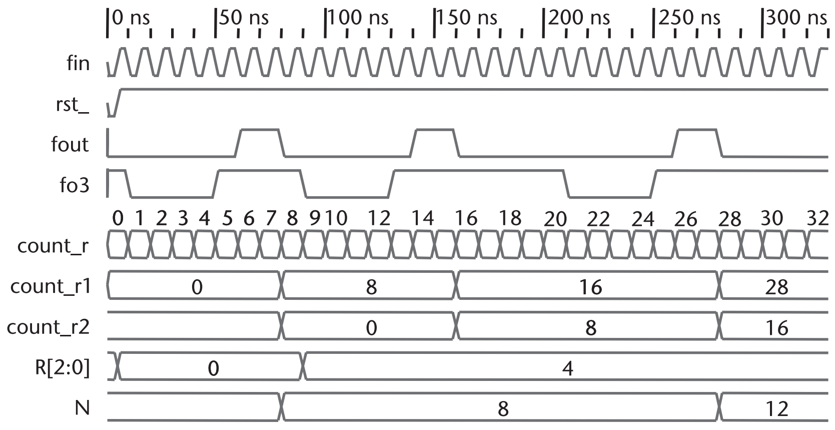


Figure 4.12 The simulated 3-bit MMD waveforms with modulus input set to R = 0 and R = 4.

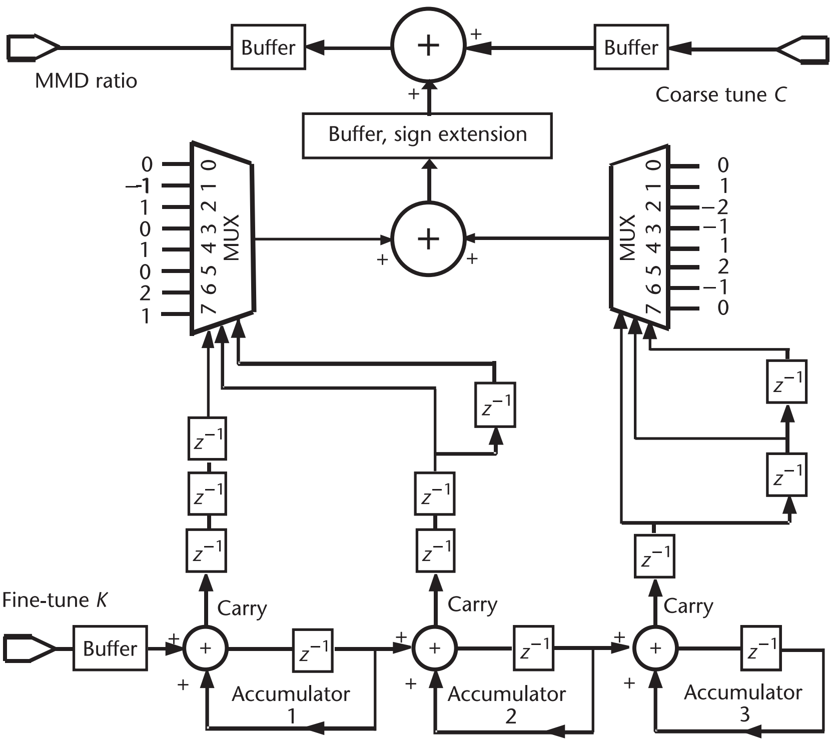


Figure 4.13 A third-order multiloop MASH ΣΔ modulator.

In this section, we provide a portion of the Verilog code for the above ΣΔ modulator design. To verify the design, the ΣΔ modulator outputs obtained in Verilog simulations are compared with the results from Matlab behavioral simulations. Bitwise agreements between Verilog and Matlab simulation results were achieved. The following Verilog code is fully synthesizable and can be easily programmed for any number of accumulator bits by using a parameterized programming style. The comments given in the code should be self-explanatory regarding the parameters and the code functions.

4.7.1 MASH ΣΔ Modulator Top-Level Structure

The programmable ΣΔ modulator can generate fractional-divider ratios by generating outputs, which vary randomly around the desired average value. Because the output is random, noise will be moved to higher frequencies (also called noise shaping) as will be discussed in Chapter 9. Thus, the first and the second accumulators form an accumulator with second-order noise shaping. The third accumulator further dithers the instantaneous division ratio in a wider range around its correct average value. The three accumulators can be selected and powered down separately to allow programmable ΣΔ architectures in the following format:

1. Only the coarse tune is selected, and all three accumulators are powered down.

2. Coarse tune and the first accumulator are selected, and the second and third accumulators are powered down.
3. Coarse tune and the first and second accumulators are selected, and the third accumulator is powered down.
4. Coarse tune and all three accumulators are selected.

```
/*
DESCRIPTION: This module models the fractional-N accumulator with
programmable MASH delta-sigma accumulators as shown in Figure 4.13.
*/
module fracds( clk,
               rst_n,
               f_in,
               k_in,
               sel_sd_in,
               sel_seed_in,
               sd_seed_in,
               seed_en1_in,
               seed_en2_in,
               seed_en3_in,
               sd_out);

parameter size = 6; //max accumulator size, parameterized coding style.

input           clk; //SD clock, namely, the reference signal.
input           rst_n; //active-low reset, _n represents active low.
input[size-1:0] f_in; //accumulator size, fractionality = F+1 = 1~64.
input[size-1:0] k_in; // numerator K of channel fine tune word K/F,
             //K = 0~63.
input[1:0]      sel_sd_in; //select the order of the ΣΔ modulator based
             //on the following rules:
  //00 --> stop clocking all accumulators, when N is an integer or
             //synthesizer is in sleep mode,
  //01 --> Use only the 1st accumulator output for fractional-N without
             //ΣΔ operation,
  //10 --> Use only the 1st and the 2nd accumulator outputs for FN with
             //2nd-order ΣΔ effect,
  //11 --> Use the 1st , 2nd and 3rd accumulator outputs for FN with
             //3rd-order ΣΔ effect.
input[1:0]      sel_seed_in; //accumulator seed value selection base on
             //the following rules:
  //00 --> all the ΣΔ accumulator is reset to zero,
  //01 --> accumulator 1 is reset to a loaded value SD_SEED, accumulator
             //2 & 3 are reset to 0,
  //10 --> accumulator 1 & 2 are reset to a loaded value SD_SEED,
             //accumulator 3 is reset to 0,
  //11 --> all 3 accumulators are reset to a loaded value SD_SEED.
input[size-1:0] sd_seed_in; //Pre-calculated start value for the ΣΔ
             //accumulator.
input           seed_en1_in, seed_en2_in, seed_en3_in;
output[3:0]     sd_out; //carry out of the three ΣΔ accumulators.

// Internal wires and buffers:
reg             seed_en1, seed_en2, seed_en3;
reg[size-1:0]   f, k, sd_seed, seed1, seed2, seed3, f2s, accum_out1,
             accum_out2, accum_out3;
reg[1:0]        sel_sd, sel_seed;
wire carry1,carry2,carry3;
reg c1_d0, c1_d1, c1_d2, c1_d3, c1_d4, c2_d0, c2_d1, c2_d2, c2_d3, c3_d0,
             c3_d1, c3_d2;
reg[3:0] sd12, sd3, sd_out;

//buffer input parameter k and f, etc.:
always @(posedge clk or negedge rst_n)
  if (!rst_n) //reset action:
```

```
  begin
    f[size-1:0] <= {size{1'b0}}; //reset to f+1 = 1
    k[size-1:0] <= {size{1'b0}};
    sel_sd[1:0] <= 2'b00;
    sel_seed[1:0] <= 2'b00;
    sd_seed[size-1:0] <= {size{1'b0}};
    seed_en1 <= 1'b0;
    seed_en2 <= 1'b0;
    seed_en3 <= 1'b0;
  end
  else
  begin
    f[size-1:0] <= f_in[size-1:0];
    k[size-1:0] <= k_in[size-1:0];
    sel_sd[1:0] <= sel_sd_in[1:0];
    sd_seed[size-1:0] <= sd_seed_in[size-1:0] + {~f_in[size-1:0]};
    sel_seed[1:0] <= sel_seed_in[1:0];
    seed_en1 <= seed_en1_in;
    seed_en2 <= seed_en2_in;
    seed_en3 <= seed_en3_in;
  end

assign f2s[size-1:0] = ~f[size-1:0]; //f2s = 2's comp of (f+1) = ~f

//generate seeds for accumulator reset:
always @ (sel_seed or sd_seed)
  begin
    case (sel_seed[1:0])
      2'b00 : begin
        seed1[size-1:0] = {size{1'b0}};
        seed2[size-1:0] = {size{1'b0}};
        seed3[size-1:0] = {size{1'b0}};
        end
      2'b01 : begin
        seed1[size-1:0] = sd_seed[size-1:0];
        seed2[size-1:0] = {size{1'b0}};
        seed3[size-1:0] = {size{1'b0}};
        end
      2'b10 : begin
        seed1[size-1:0] = sd_seed[size-1:0];
        seed2[size-1:0] = sd_seed[size-1:0];
        seed3[size-1:0] = {size{1'b0}};
        end
      2'b11 : begin
        seed1[size-1:0] = sd_seed[size-1:0];
        seed2[size-1:0] = sd_seed[size-1:0];
        seed3[size-1:0] = sd_seed[size-1:0];
        end
      default:begin
        seed1[size-1:0] = {size{1'b0}};
        seed2[size-1:0] = {size{1'b0}};
        seed3[size-1:0] = {size{1'b0}};
        end
    endcase
  end

//Instantiate accumulators 1, 2, and 3:
accum accum1( .clk(clk),
       .rst_n(rst_n),
       .f(f),
       .f2s(f2s),
       .accum_in(k),
       .seed(seed1),
       .seed_en(seed_en1),
       .carry(carry1),
       .accum_out(accum_out1) );
```

```
accum accum2( .clk(clk),
        .rst_n(rst_n),
        .f(f),
        .f2s(f2s),
        .accum_in(accum_out1),
        .seed(seed2),
        .seed_en(seed_en2),
        .carry(carry2),
        .accum_out(accum_out2) );

accum accum3( .clk(clk),
        .rst_n(rst_n),
        .f(f),
        .f2s(f2s),
        .accum_in(accum_out2),
        .seed(seed3),
        .seed_en(seed_en3),
        .carry(carry3),
        .accum_out(accum_out3) );

//select sd and carry:
always @ (sel_sd or carry1 or carry2 or carry3)
  begin
    case (sel_sd[1:0])
      2'b00 : begin
        c1_d0=1'b0;
        c2_d0=1'b0;
        c3_d0=1'b0;
        end
      2'b01 : begin
        c1_d0=carry1;
        c2_d0=1'b0;
        c3_d0=1'b0;
        end
      2'b10 : begin
        c1_d0=carry1;
        c2_d0=carry2;
        c3_d0=1'b0;
        end
      2'b11 : begin
        c1_d0=carry1;
        c2_d0=carry2;
        c3_d0=carry3;
        end
      default:begin
        c1_d0=carry1;
        c2_d0=carry2;
        c3_d0=carry3;
        end
    endcase
  end

//MASH sigma-delta delays as shown in Figure 4.13:
always @(posedge clk or negedge rst_n)
  if (!rst_n)
  begin
    c1_d1 <= 1'b0;
    c1_d2 <= 1'b0;
    c1_d3 <= 1'b0;
    c1_d4 <= 1'b0;
    c2_d1 <= 1'b0;
    c2_d2 <= 1'b0;
    c2_d3 <= 1'b0;
    c3_d1 <= 1'b0;
    c3_d2 <= 1'b0;
  end
  else
```

```
    begin
      c1_d4 <= c1_d3;
      c1_d3 <= c1_d2;
      c1_d2 <= c1_d1;
      c1_d1 <= c1_d0;

      c2_d3 <= c2_d2;
      c2_d2 <= c2_d1;
      c2_d1 <= c2_d0;

      c3_d2 <= c3_d1;
      c3_d1 <= c3_d0;
    end

  //1st- and 2nd-order MASH sigma-delta coefficients as shown in Figure 4.13:
  always @ (c1_d4 or c2_d2 or c2_d3)
    begin
      case ({c1_d4,c2_d2,c2_d3})
        3'b000 : sd12=4'b0000;
        3'b001 : sd12=4'b1111;
        3'b010 : sd12=4'b0001;
        3'b011 : sd12=4'b0000;
        3'b100 : sd12=4'b0001;
        3'b101 : sd12=4'b0000;
        3'b110 : sd12=4'b0010;
        3'b111 : sd12=4'b0001;
        default: sd12=4'b0000;
      endcase
    end

  //3rd-order MASH sigma-delta coefficients as shown in Figure 4.13:
  always @ (c3_d0 or c3_d1 or c3_d2)
    begin
      case ({c3_d0, c3_d1, c3_d2})
        3'b000 : sd3=4'b0000;
        3'b001 : sd3=4'b0001;
        3'b010 : sd3=4'b1110;
        3'b011 : sd3=4'b1111;
        3'b100 : sd3=4'b0001;
        3'b101 : sd3=4'b0010;
        3'b110 : sd3=4'b1111;
        3'b111 : sd3=4'b0000;
        default: sd3=4'b0000;
      endcase
    end

  //MASH sigma-delta output:
  always @(posedge clk or negedge rst_n)
    if (!rst_n)
    begin
      sd_out[3:0] <= 4'b0000;
    end
    else
    begin
      sd_out[3:0] <= sd12[3:0] + sd3[3:0];
    end

  endmodule //close fracds
```

4.7.2 Fractional Accumulator with Programmable Size and Seed-Loading Capability

The presented $\Sigma\Delta$ accumulator has a special feature of loading precalculated start values to three accumulators to avoid artificial spur generation. It is known that any repetition of a time sequence will cause artificial spurs in the spectrum with a frequency inversely proportional to the repetition's period. Loading different start values (seeds) to different accumulators will break the repetition in the time domain. Two control bits (SEL_SEED) were designed to load the seeds flexibly as follows:

1. If SEL_SEED = 00, all the ΣΔ accumulators are reset to zero.
2. If SEL_SEED = 01, the first accumulator is reset to a precalculated seed, and the second and the third accumulators are reset to zero.
3. If SEL_SEED = 10, the first and the second accumulators are reset to a precalculated seed, and the third accumulator is reset to zero.
4. If SEL_SEED = 11, all the three accumulators are reset to a loaded seed value.

```
/*
DESCRIPTION: This module models the fractional-N accumulator with
programmable size as shown in Figure 4.13.
*/

module accum( clk,
              rst_n,
              f,
              f2s,
              accum_in,
              seed,
              seed_en,
              carry,
              accum_out);

parameter size = 6; //max accumulator size

input           clk;
input           rst_n; //active-low reset
input[size-1:0] f; //fractionality = f+1 = 1~64
input[size-1:0] f2s; //2's comp of fractional accumulator size (f+1)
input[size-1:0] accum_in; //accumulator input.
input[size-1:0] seed; //precalculated start value for the accumulator.
input           seed_en;
output          carry; //buffered carry out.
output[size-1:0] accum_out; //buffered accumulator output.

reg[size-1:0]   accum_out; //buffered accumulator output.
reg[size:0]     accum; //sign extended to size+1 bits

//programmable accumulator:
always @(posedge clk or negedge rst_n)
  if (!rst_n)
  begin
    accum[size:0] <= {(size+1){1'b0}}; //reset to 0.
  end
  else if(seed_en)
  begin
    accum[size:0] <= {1'b0,seed[size-1:0]}; //load seed.
  end
  else
  begin
    if(accum[size]) //normal operation with carry out
    begin
      accum[size:0] <= {1'b0,accum_in[size-1:0]} + {1'b0,accum[size-1:0]}
              {1'b0,f2s[size-1:0]};
    end
    else //normal operation without carry out
    begin
      accum[size:0] <= {1'b0,accum_in[size-1:0]} + {1'b0,accum[size-1:0]};
    end
  end

//buffer accumulator output:
always @(posedge clk or negedge rst_n)
  if (!rst_n)
  begin
    accum_out[size-1:0] <= {size{1'b0}}; //reset to 0.
  end
  else if(accum[size]) //normal operation with carry out
    begin
      accum_out[size-1:0] <= accum[size-1:0];
    end
  else //normal operation without carry out
  begin
    accum_out[size-1:0] <= accum[size-1:0] + f[size-1:0] + {{(size-1)
            {1'b0}},1'b1};
  end

assign carry = accum[size];

endmodule //close accum
```

4.7.3 Reset Synchronization

The presented ΣΔ modulator also features an active-high soft reset. The soft reset differs from the global reset in the sense that the global reset signal resets every flip-flop to zero, while the soft reset signal only resets the accumulators with desired frequency words and seed values. The release of asynchronous reset is synchronized to the clock as illustrated in Figure 4.14, where *tm* is the test mode signal to enable the internal scan for digital block testing. During scan testing, the internal generated reset signals and their synchronizations will be disabled.

```
/*
DESCRIPTION: This module models asynchronous reset with synchronous
release as shown in Figure 4.14.
*/

module rst_syn (clk,
         rst_in_n,
         rst_out_n,
         tm);

input    clk;
input    rst_in_n; //input global asynchronous rst
output   rst_out_n; //output synchronized rst
input    tm; //active-high test mode
```

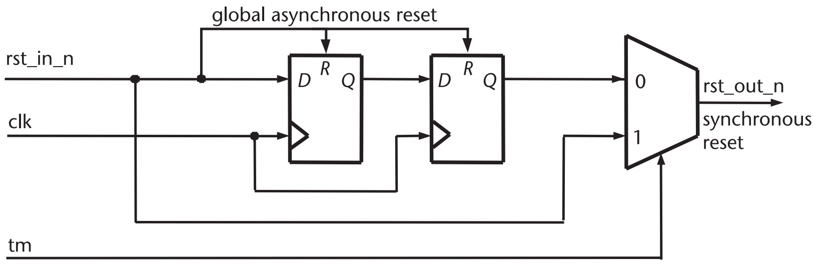


Figure 4.14 The reset synchronization for ΣΔ accumulators.

```
reg rst1, rst2;

always @(posedge clk or negedge rst_in_n)
  if(!rst_in_n)
  begin
    rst1 <= 1’b0;
    rst2 <= 1’b0;
  end
  else
  begin
    rst2 <= rst1;
    rst1 <= rst_in_n;
  end

assign rst_out_n = tm ? rst_in_n : rst2;

endmodule //end rst_syn
```

4.7.4 Simulated Results

Figure 4.15 gives Verilog simulation results of the ΣΔ output and its spectrum for the case where coarse tune $C = 17$, fractionality $F = 30$, and fine tune $K = 27$. All three accumulator outputs are selected, and the fractional spurs are thus shaped by a third-order MASH ΣΔ fractional accumulator. Figure 4.15(a) shows how the third-order ΣΔ dithers the division ratio around its average value ($C = 17$) in the time domain. Figure 4.15(b) clearly demonstrates the noise-shaping effect in the frequency domain. Because this is a third-order loop (three accumulators), the noise shaping is third order, which means the spectrum of noise rises at 60 dB/dec as expected.

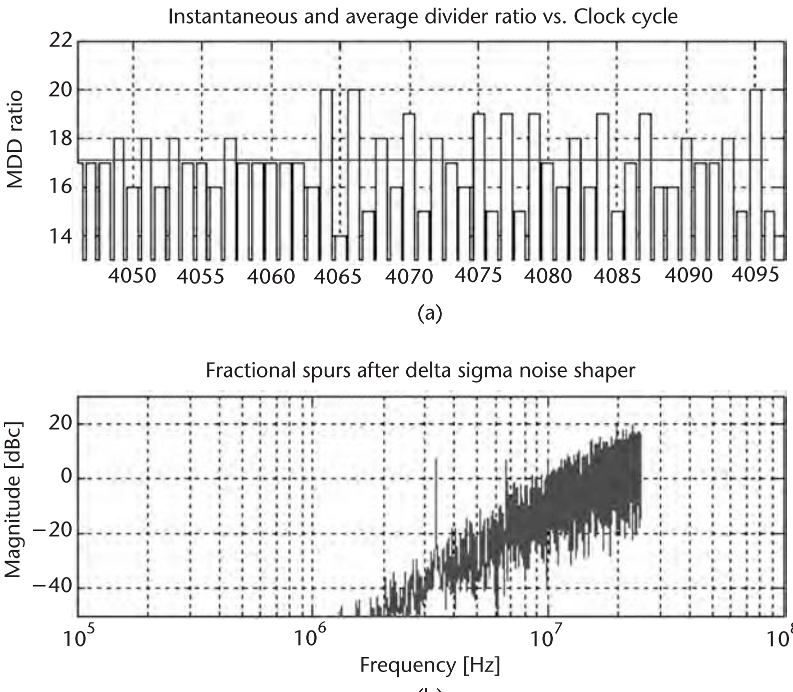


Figure 4.15 (a) Simulated division ratio, and (b) its spectrum.

References

[1] Ciletti, M. D., *Advanced Digital Design with the Verilog HDL*, Upper Saddle River, NJ: Pearson Education, 2003.

[2] Lee, S., *Advanced Digital Logic Design Using Verilog, State Machines, and Synthesis for FPGAs*, Washington, DC: Thomson Publishing Group, 2003.

[3] Moorby, P. R., and D. E. Thomas, *The Verilog Hardware Description Language*, 4th ed., Dordrecht, the Netherlands: Kluwer Academic Publishers, 1998.

[4] Palnitkar, S., *Verilog HDL*, 2nd ed., Upper Saddle River, NJ: Prentice Hall, 2003.

[5] Bhasker, J. A., *Verilog HDL Primer*, Allentown, PA: Star Galaxy Press, 1997.

[6] Sternheim, E., et al., *Digital Design and Synthesis with Verilog HDL*, San Jose, CA: Automata Publishing Co., 1993.

[7] Vaucher, C. S., et al., "A Family of Low-Power Truly Modular Programmable Dividers in Standard 0.35 μm CMOS Technology," *IEEE J. Solid-State Circuits*, Vol. 35, July 2000, pp. 1039–1045.

[8] Rogers, J. W. M., et al., "A Fully Integrated Multi-Band $\Sigma\Delta$ Fractional-*N* Frequency Synthesizer for a MIMO WLAN Transceiver RFIC," *IEEE J. Solid-State Circuits*, Vol. 40, No. 3, March, 2005, pp. 678–689.

CHAPTER 5

CMOS Logic and Current Mode Logic

5.1 Introduction

This chapter describes basic logic circuits. This can be seen as largely background material for applications in later chapters in the design of dividers and phase detectors in frequency synthesizers. The types of logic discussed will be CMOS rail-to-rail logic, CMOS current mode logic (CML), bipolar CML, and bipolar emitter coupled logic (ECL). Note that CML is a general term and applies to both bipolar and CMOS; however, with metal oxide semiconductor (MOS) transistors, it is often called MOS current mode logic (MCML). ECL is the name often given to bipolar CML that has emitter followers as the output stage transistors. At low frequencies, CMOS rail-to-rail is preferred for its simplicity and low static power dissipation, while, at higher frequencies, CML or ECL is used, as they can operate faster with lower power because of the reduced output swing. As shown in Figure 5.1, when it is not switching, CMOS rail-to-rail logic does not consume any current, while CML does. CMOS rail-to-rail logic consumes current only during transitions, and its power consumption is proportional to the operation frequency. CML bias current must rise as the speed of switching increases, just as CMOS rail-to-rail logic does, but it does so at a slower rate. Thus, above some frequency, CML becomes a lower-power solution. CMOS rail-to-rail logic is differential and, therefore, has good power-supply rejection, which is preferred in many synthesizer applications. Various types of logic and some important design trade-offs are summarized in Table 5.1.

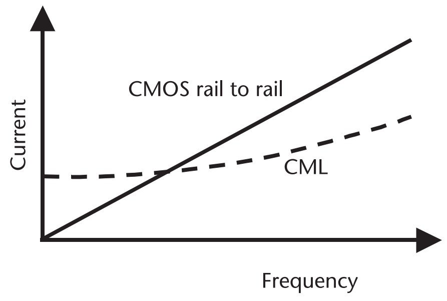


Figure 5.1 Comparison of current for CMOS rail-to-rail and CML logic versus frequency.

Table 5.1 Comparison of Different Logic Styles

Logic Type	*Noise Performance*	*Power-Supply Rejection*	*Maximum Speed*	*Power Dissipation*
CMOS rail-to-rail	Bad	Bad	Moderate	Low at low frequency, high at high frequency
CMOS CML	Good	Good	High	High at low frequency, low at high frequency
Bipolar CML, ECL	Excellent	Good	Very High	High at low frequency, low at high frequency

5.2 CMOS Logic Circuits

CMOS rail-to-rail logic is by far the most commonly used type of logic circuit; however, in synthesizer design, often CML is preferred. The bulk of this chapter will focus on CML; however, for lower-speed applications, CMOS rail-to-rail logic is often used. CMOS rail-to-rail logic has outputs that are either at or very close to one of the power supplies. Since the voltage in this case must change by a large amount and, hence, requires larger charge and discharge time, rail-to-rail logic is often slower than the other types of logic we will use. Also, even though the dc power consumption of CMOS rail-to-rail logic is zero, when such circuits are switched at high speed, they can consume a lot of power (just check the heat sink on your Pentium!).

Basic CMOS rail-to-rail logic functions are shown in Figure 5.2 [1, 2]. They are always made from a pull-up and pull-down network to pull the output to one rail or the other, depending on the inputs. If the transistors in the figure are thought of as switches, it is not hard to see how these circuits implement their various logic functions. Their speed is largely determined by how much capacitance they have to drive and how much current they can source or sink. The transistors in these circuits are usually sized by choosing width W and length L so that the pull-down and pull-up currents are equal. For example, transistors M_5 and M_6 will be able to sink about four times as much current as transistors M_3 and M_4 if all transistors are the same size and p-channel MOS (PMOS) and n-channel MOS (NMOS) are matched. This can be seen by assuming that all transistors have an on resistance given by

$$r_{\text{on}} = R\left(\frac{L}{W}\right) \tag{5.1}$$

where R is a constant. Also note that the pull-down and pull-up currents will vary if there is more than one transistor in parallel. Thus, for instance, in Figure 5.2(b) the pull-down current will be twice as much if both A and B are high versus what it would be if only A or B were high by itself.

The power dissipation of CMOS rail-to-rail logic can also be determined quite easily. If the CMOS gate is driving a load capacitance C_L, then, each cycle, the capacitor must be charged up to V_{DD} and then discharged to V_{SS}. Each time this happens, the energy dissipated is

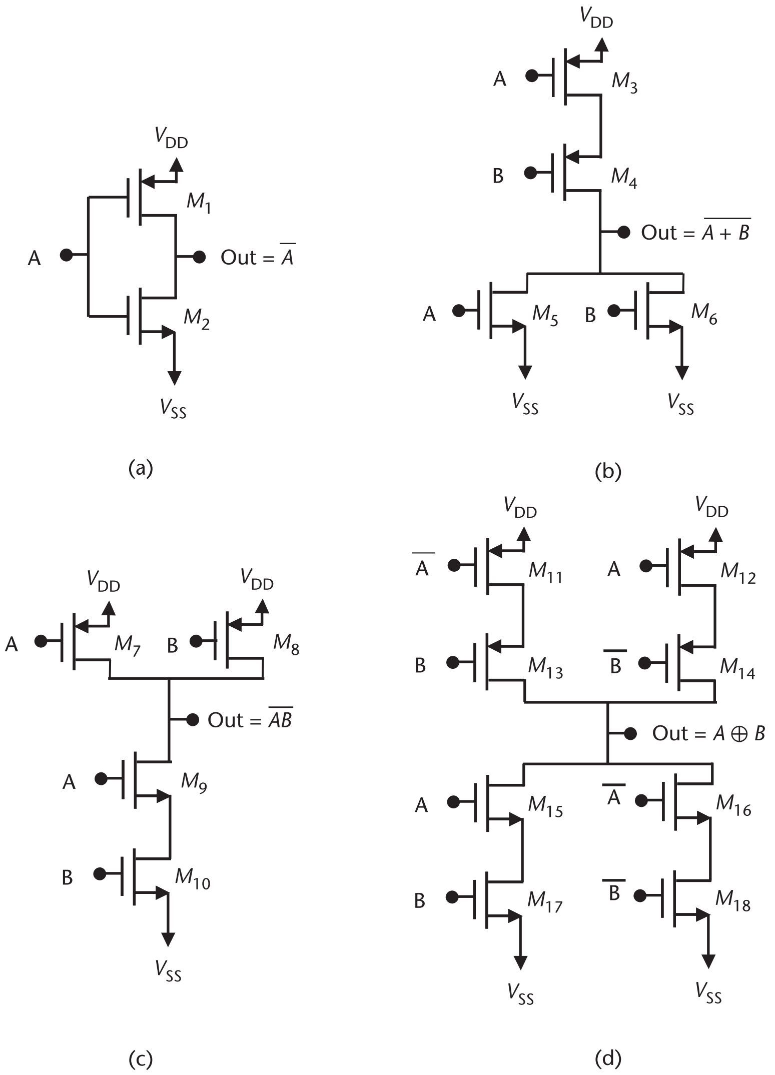


Figure 5.2 CMOS digital logic gates: (a) inverter, (b) NOR gate, (c) NAND gate, and (d) XOR gate.

$$E = \frac{C_L(V_{\mathrm{DD}} - V_{\mathrm{SS}})^2}{2} \tag{5.2}$$

Since both charging and discharging results in this dissipation, coming if the clock frequency is f, then the power dissipated in the gate is

$$P = C_L(V_{\mathrm{DD}} - V_{\mathrm{SS}})^2 f \tag{5.3}$$

5.3 Large-Signal Behavior of Bipolar and CMOS Differential Pairs

In CML, the dc current is constant, which leads to less switching noise. CML circuits are intrinsically differential, making interfaces with the analog parts of the synthesizer easier as these blocks are often also differential for a variety of reasons. The basis for all CML is the differential pair [3]. It is important to switch the pair in digital applications, which means that a large enough differential voltage must be applied to the input. For the bipolar and CMOS cases, respectively, shown in Figure 5.3, the tail current as a function of either drain or collector currents can be written as

$$I_{\mathrm{EE}} = i_{C1} + i_{C2} \text{ or } I_{\mathrm{EE}} = i_{D1} + i_{D2} \tag{5.4}$$

Also note that the input voltages can be written as the sum of base-emitter or gate-source voltages:

$$v_1 = v_{\mathrm{BE1}} - v_{\mathrm{BE2}} \text{ or } v_1 = v_{\mathrm{GS1}} - v_{\mathrm{GS2}} \tag{5.5}$$

We will need an expression for the collector or drain current as a function of the input voltage. We start, first, with the bipolar case, as it is simpler. The voltage current relationship for a bipolar transistor is

$$i_C = I_S e^{\frac{v_{\mathrm{BE}}}{v_T}} \tag{5.6}$$

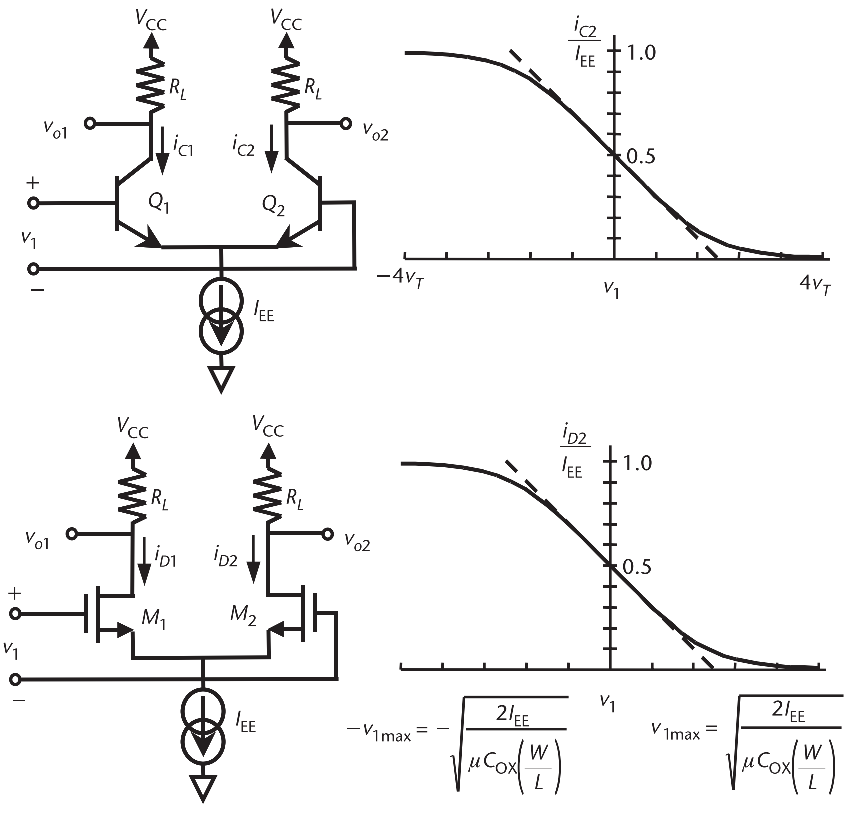


Figure 5.3 Differential pairs in CMOS and bipolar technology.

where I_S is the transistor reverse saturation current, and v_T is thermal voltage, kT/q, which is about 25 mV at room temperature. This can be written as

$$v_{\mathrm{BE}} = v_T \ln\left(\frac{i_C}{I_S}\right) \tag{5.7}$$

Therefore, (5.5) can be rewritten as

$$v_1 = v_T \ln\left(\frac{i_{C1}}{I_S}\right) - v_T \ln\left(\frac{i_{C2}}{I_S}\right) \tag{5.8}$$

Now, making use of (5.4),

$$v_1 = v_T \ln\left(\frac{i_{C1}}{I_S}\right) - v_T \ln\left(\frac{I_{\mathrm{EE}} - i_{C1}}{I_S}\right) \tag{5.9}$$

After some manipulation,

$$e^{\frac{v_1}{v_T}} = \frac{i_{C1}}{I_{\mathrm{EE}} - i_{C1}} \tag{5.10}$$

$$i_{C1} = I_{\mathrm{EE}}\left(\frac{e^{\frac{v_1}{v_T}}}{1 + e^{\frac{v_1}{v_T}}}\right)$$

i_{C2} can be solved in a similar way:

$$i_{C2} = I_{\mathrm{EE}}\left(\frac{e^{\frac{-v_1}{v_T}}}{1 + e^{\frac{-v_1}{v_T}}}\right) \tag{5.11}$$

Thus, the bipolar differential pair is completely switched when v_1 is approximately $4v_T$ or larger, regardless of the size of the transistor used or the current. Therefore, for maximum speed and minimum capacitance, minimum geometry devices are preferred, provided noise is not an issue, and the current is not beyond the peak f_T current density.

Equations for the CMOS differential pair can be solved in a similar manner, although a few more steps are required. The simple square law voltage-current relationship for a CMOS transistor is

$$i_D = \frac{\mu C_{\mathrm{ox}}}{2}\left(\frac{W}{L}\right)(v_{GS} - V_T)^2 \tag{5.12}$$

which can be rewritten as

$$v_{GS} = \sqrt{\frac{2}{\mu C_{\mathrm{ox}}}\left(\frac{L}{W}\right)}\sqrt{i_D} + V_T \tag{5.13}$$

Therefore, (5.5) can be rewritten as

$$v_1 = \sqrt{\frac{2}{\mu C_{\mathrm{ox}}}\left(\frac{L}{W}\right)}\left(\sqrt{i_{D1}} - \sqrt{i_{D2}}\right) \tag{5.14}$$

Now, making use of (5.4),

$$v_1 = \sqrt{\frac{2}{\mu C_{\mathrm{ox}}}\left(\frac{L}{W}\right)}\left(\sqrt{i_{D1}} - \sqrt{I_{\mathrm{EE}} - i_{D1}}\right) \tag{5.15}$$

Now, squaring both sides of (5.15) gives

$$v_1^2 = \frac{2}{\mu C_{\mathrm{ox}}}\left(\frac{L}{W}\right)\left(i_{D1} - 2\sqrt{i_{D1}}\sqrt{I_{\mathrm{EE}} - i_{D1}} + I_{\mathrm{EE}} - i_{D1}\right) \tag{5.16}$$

$$v_1^2 = \frac{2}{\mu C_{\mathrm{ox}}}\left(\frac{L}{W}\right)\left(I_{\mathrm{EE}} - 2\sqrt{i_{D1}}\sqrt{I_{\mathrm{EE}} - i_{D1}}\right)$$

Moving all terms with i_{D1} in them to one side yields

$$\frac{4}{\mu C_{\mathrm{ox}}}\left(\frac{L}{W}\right)\sqrt{i_{D1}}\sqrt{I_{\mathrm{EE}} - i_{D1}} = \frac{2}{\mu C_{\mathrm{ox}}}\left(\frac{L}{W}\right)I_{\mathrm{EE}} - v_1^2 \tag{5.17}$$

Squaring, to remove the roots, results in

$$\frac{16}{(\mu C_{\mathrm{ox}})^2}\left(\frac{L}{W}\right)^2\left(I_{\mathrm{EE}}i_{D1} - i_{D1}^2\right) = \frac{4}{(\mu C_{\mathrm{ox}})^2}\left(\frac{L}{W}\right)^2 I_{\mathrm{EE}}^2 - \frac{4}{\mu C_{\mathrm{ox}}}\left(\frac{L}{W}\right)I_{\mathrm{EE}}v_1^2 + v_1^4 \tag{5.18}$$

Collecting terms gives

$$\begin{aligned}&\frac{16}{(\mu C_{\mathrm{ox}})^2}\left(\frac{L}{W}\right)^2 i_{D1}^2 \\ &\quad - \frac{16}{(\mu C_{\mathrm{ox}})^2}\left(\frac{L}{W}\right)^2 I_{\mathrm{EE}}i_{D1} + \frac{4}{(\mu C_{\mathrm{ox}})^2}\left(\frac{L}{W}\right)^2 I_{\mathrm{EE}}^2 - \frac{4}{\mu C_{\mathrm{ox}}}\left(\frac{L}{W}\right)I_{\mathrm{EE}}v_1^2 + v_1^4 = 0 \\ &\qquad i_{D1}^2 - I_{\mathrm{EE}}i_{D1} + \frac{1}{4}I_{\mathrm{EE}}^2 - \frac{\mu C_{\mathrm{ox}}}{4}\left(\frac{W}{L}\right)I_{\mathrm{EE}}v_1^2 + \frac{(\mu C_{\mathrm{ox}})^2}{16}\left(\frac{W}{L}\right)^2 v_1^4 = 0\end{aligned} \tag{5.19}$$

This can be solved for i_{D1}:

$$i_{D1} = \frac{I_{\text{EE}}}{2}\left[1 \pm \sqrt{v_1^2 \frac{\mu C_{\text{ox}}}{I_{\text{EE}}}\left(\frac{W}{L}\right) - \frac{(\mu C_{\text{ox}})^2}{4I_{\text{EE}}^2}\left(\frac{W}{L}\right)^2 v_1^4}\,\right] \tag{5.20}$$

The term inside the brackets will have a peak value of two at some input voltage of $v_{1\max}$. This voltage can be determined by setting the derivative of (5.20) to zero, and the result is given by

$$v_{1\,\max} = \sqrt{\frac{2I_{\text{EE}}}{\mu C_{\text{ox}}\left(\frac{W}{L}\right)}} \tag{5.21}$$

The current becomes

$$i_{D1} = \frac{I_{\text{EE}}}{2}\left(1 \pm \sqrt{2-1}\right) = \frac{I_{\text{EE}}}{2}(1 \pm 1) = 0,\ I_{\text{EE}} \tag{5.22}$$

Clearly, (5.20) is no longer valid for values greater than $v_{1\max}$, as the equation then incorrectly predicts that the current starts to decrease again. For larger values of voltage, one side continues to take all the current, and the other side just becomes more firmly off. In real circuits, for large v_1, the source voltage then starts to follow input voltage, limiting the total voltage to $v_{1\max}$.

Note that in (5.20) the "+" sign is correct for v_1 greater than zero, and the "–" sign is correct for v_1 less than zero. Thus, the current moves around the quiescent point of $I_{\text{EE}}/2$. In this case, complete switching is dependent on current and device size. The current is completely switched when the term under the square root in (5.20) is one.

From (5.21), it can be seen that for larger current, the required switching voltage increases, while for larger W/L ratios, the switching voltage decreases. Since the switching voltage is inversely related to the square root of W/L, large increases of W/L are often required for a particular decrease in switching voltage. This can make it difficult to reduce switching voltage by this method. As an example, suppose we have a 0.1-μm technology. Then, for a width of 1 μm, the switching voltage (normalized to $\sqrt{2I_{\text{EE}}/\mu C_{\text{ox}}}$) is 0.316; for a width of 10 μm, it is 0.1; and for a width of 100 μm, it is 0.0316. Thus, beyond 10 μm, transistor size has to become large by IC standards to have much effect on switching voltage. For CML, a 10-μm transistor would already be quite large and require a large current to switch quickly. Since speed is often an issue, using huge transistors is not a very practical solution.

5.4 Effect of Capacitance on Slew Rate

The bias current through a differential pair required for proper operation at a given speed is dependent on the load capacitance as well as the required output swing. If there were no capacitance, then arbitrarily large output swings could be achieved with arbitrarily small currents. Of course, this is unrealistic. When the differential pair is fully switched, one side has no current such that the output voltage on this side is at the rail (V_{CC}), while the other side has all the current flowing through it. Thus, the peak differential swing is

$$v_{o1} = V_{CC} - I_{EE} R_L \tag{5.23}$$

Generally, the swing must be large enough to fully switch the following stage that is being driven. Any additional swing is a waste of power. However, as output waveforms are not perfectly square, to switch faster, additional swing will be required to ensure that the fully switched condition is satisfied over a longer period of time. In the case of bipolar, 100–200-mV peak is typically considered to be sufficient. In CMOS, usually more is required, depending on the current and transistor sizes involved, but generally 200–400-mV peak is a good ballpark number.

The load capacitance will determine the slew rate of the stage. The slew rate is the rate of change of the output voltage. For a capacitance C, the rate of change of voltage is given by

$$\frac{dV}{dt} = \frac{1}{C} \cdot \frac{dq}{dt} \tag{5.24}$$

where q is the charge on the capacitor. Assuming that each side of the differential pair has a load capacitance C_L connected to it, the slew rate is given by

$$\text{Slew rate} = \frac{dv_{o1}}{dt} = \frac{1}{C_L} \cdot I_{\text{cap}} \tag{5.25}$$

where I_{cap} is the current flowing through the capacitor. The time required for the output to fully switch should be a small percentage of the period of the square wave. Otherwise, the wave will have a triangular shape, as shown in Figure 5.4.

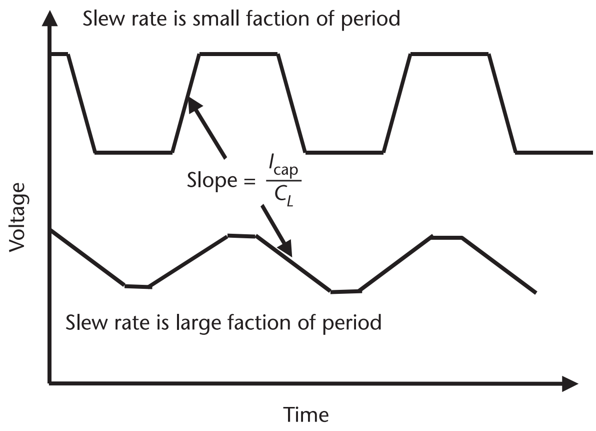

Slew rate is small faction of period
Slope = $\frac{I_{cap}}{C_L}$
Slew rate is large faction of period
Voltage
Time

Figure 5.4 Illustration of the effect of slew rate on a square wave.

Now the total transistor current I_D is divided between the capacitor current I_{cap} and the load resistor current I_R such that

$$I_D = I_{\text{cap}} + I_R \tag{5.26}$$

From (5.25) and making use of Ohm's law, this can be rewritten as

$$I_D = C_L \frac{dv_{o1}}{dt} + \frac{v_{o1}}{R_L} \tag{5.27}$$

An important consideration is that the transistor current does not switch instantaneously. This can lead to a lower slew rate than would otherwise be expected from the equations above. Assume that the stage is being switched by another stage that is also loaded by $R_L C_L$. With such loading, the bias current does not switch on or off instantaneously; rather, it has an exponential component to it. In order to determine output voltages and currents, (5.27) must be modified to include a changing bias current:

$$I_D(t) = I_{\text{EE}}\left(1 - e^{\frac{-t}{R_L C_L}}\right) = C_L \frac{dv_{o1}}{dt} + \frac{v_{o1}}{R_L} \tag{5.28}$$

Solving this differential equation yields (for the case of current turning on)

$$v_{o1}(t) = I_{\text{EE}} R_L \left(1 - e^{-\frac{t}{R_L C_L}} - \frac{t}{R_L C_L} e^{-\frac{t}{R_L C_L}}\right) \tag{5.29}$$

In the case of current turning off, it yields

$$v_{o1}(t) = I_{\text{EE}} R_L \left(e^{-\frac{t}{R_L C_L}} + \frac{t}{R_L C_L} e^{-\frac{t}{R_L C_L}}\right) \tag{5.30}$$

The instantaneous current through the capacitor is the difference between the total current and the current in the resistor:

$$I_{\text{cap}}(t) = I_{\text{D}}(t) - \frac{v_{o1}(t)}{R_L} = \pm I_{\text{EE}} \cdot \frac{t}{R_L C_L} \cdot e^{-\frac{t}{R_L C_L}} \tag{5.31}$$

for either turning on or turning off the transistor current. Equations (5.31), (5.30), and (5.29) are plotted in Figure 5.5. One can see that, with the finite turn-on time of transistor current, the maximum capacitor current is about 38% of I_{EE}. If the transistor had switched current instantaneously to I_{EE}, the peak capacitor current would also have been I_{EE} at the start of the transient.

Thus, a rough estimate for the time to switch logic levels through a circuit made of CML inverters would be

$$\text{Time to switch} = 5 \cdot R_L C_L \tag{5.32}$$

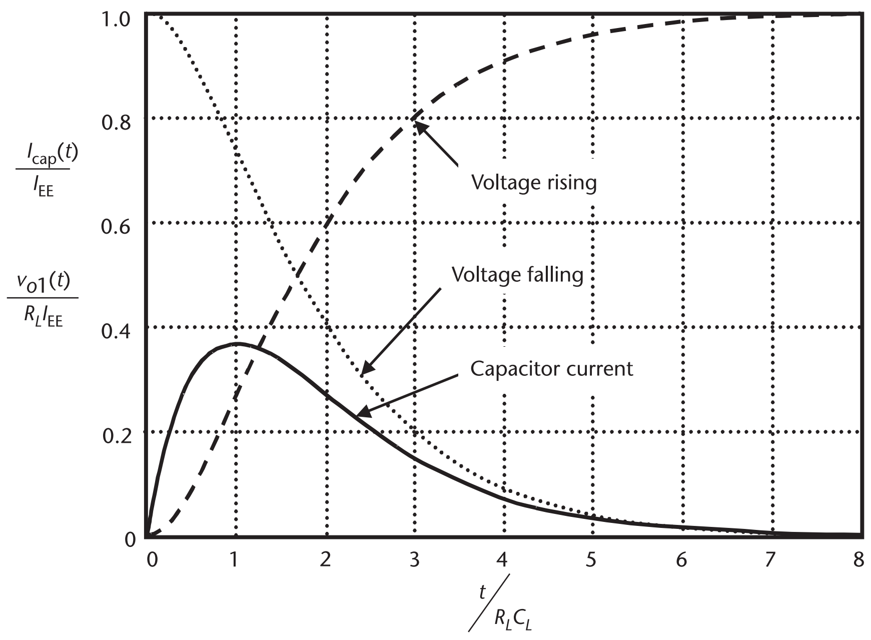


Figure 5.5 Illustration of the effect of slew rate on a square wave.

However, since the following circuit will start to switch when its input voltage is halfway to its final value, the time required (often simply called delay) is approximately

$$\text{Delay} = 1.5 \cdot R_L C_L \tag{5.33}$$

Thus, for n stages, the time between the first stage's starting to switch and the nth stage's starting to switch is

$$\text{Delay}_n = 1.5 \cdot R_L C_L \cdot n \tag{5.34}$$

For the circuit to reach maximum swing, the time to switch cannot be more than one-half of a period. Therefore, from (5.32), the maximum frequency at which a circuit could operate would be

$$f_{\max} = \frac{1}{10 \cdot R_L C_L} \tag{5.35}$$

The instantaneous slew rate as a function of time is

$$\text{Slew rate} = I_{EE} \cdot \frac{t}{R_L C_L^2} \cdot e^{-\frac{t}{R_L C_L}} \tag{5.36}$$

Since slew rate is often useful for doing quick hand calculations, a quick estimate of the maximum slew rate is based on an estimate of $I_{EE}/3$ as the maximum capacitor current shown in Figure 5.5. The resulting slew rate is

$$\text{Slew rate} = \frac{I_{\text{EE}}}{3C_L} \tag{5.37}$$

5.5 Trade-Off Between Power Consumption and Speed

In general, with CML circuits, more current results in faster operation. This is a basic trade-off between current, load resistance, and capacitance. In bipolar CML, this relationship is simple because the transistors do not have to scale with changing bias current. If the current in a bipolar CML inverter is doubled, the load resistance will be cut in half for the same output swing. Since current is doubled, but the capacitance remains the same, the slew rate of the circuit is also doubled. With CMOS, the effect of changing bias current is somewhat more complicated since the size of the transistors also needs to be adjusted to switch with a particular voltage swing. For a given desired output swing v_o, the required transistor size for a given current has previously been given as

$$v_o = \sqrt{\frac{2I_{\text{EE}}}{\mu C_{\text{ox}}\left(\frac{W}{L}\right)}} \tag{5.38}$$

If the current were doubled, then to keep the switching voltage constant, the transistor W/L ratio would also have to be doubled. Therefore, the transistor size would double and so would its capacitance, keeping the slew rate roughly constant. Thus, it might seem that speed is independent of current. If this were the case, one would always use minimum current; however, not all capacitance in the circuit will double when the current is doubled. For example, one would expect the interconnect capacitance to remain relatively constant, even when the transistor size is doubled. Therefore, even with CMOS, higher currents do, in general, make for faster circuits.

Example 5.1: Speed Versus Current Trade-Off in CML Inverters

Determine the delay through a set of four series-connected CML inverters (differential amplifiers as shown in Figure 5.3) in 0.18-μm CMOS technology. Design the inverters to operate with a 1.8-V supply with a 500-mV peak-to-peak voltage swing, and set all the tail currents to the same value, first 100 μA, then 1 mA, and, finally, 10 mA. Assume that each circuit node has 20 fF of parasitic interconnect capacitance loading it and, for this process, that $\mu C_{\text{ox}} = 215\ \mu\text{A/V}^2$ and $C_{\text{ox}} = 6.5\ \text{fF}/\mu\text{m}^2$.

Solution: The first step is to determine the required transistor size to switch with a 500-mV peak signal. Since the waveforms should look approximately square, some margin should be allowed, so 300 mV for full switching will be used. Assuming a length of 0.18 μm for the transistors for maximum speed, the widths of the devices can be determined from (5.38) as

$$W = \frac{2I_{\text{EE}}L}{\mu C_{\text{ox}} v_o^2} = 1.9\ \mu\text{m},\ 19\ \mu\text{m},\ 190\ \mu\text{m}$$

for the three currents. The load resistance for each current must give a voltage drop of 500 mV when all of the tail current is being drawn through it. Therefore,

$$R_L = \frac{v_o}{I_{\mathrm{EE}}} = 5\ \mathrm{k}\Omega,\ 500\Omega,\ 50\Omega$$

for the three currents.

Now, for each of these currents, the C_{gs} for the transistors can also be worked out as

$$C_{\mathrm{gs}} = WLC_{\mathrm{ox}} = 2.2\ \mathrm{fF},\ 22\ \mathrm{fF},\ 222\ \mathrm{fF}$$

Thus, the total load capacitance for the three currents will be 22.2 fF, 42 fF, and 242 fF. Consequently, the time to switch through a stage for each of the three currents will be

$$\begin{aligned}\text{Time to switch} &= 5 \cdot RC_{\mathrm{total}} \\ &= 5 \cdot 5\ \mathrm{k}\Omega(22.2\ \mathrm{fF}),\ 5 \cdot 500\Omega(42\ \mathrm{fF}),\ 5 \cdot 50\Omega(242.2\ \mathrm{fF}) \\ &= 555\ \mathrm{ps},\ 105\ \mathrm{ps},\ 60.5\ \mathrm{ps}\end{aligned} \tag{5.39}$$

From this simple analysis, it can be seen that, if the parasitic capacitance is larger than the transistor capacitance, then more current is advantageous. However, if C_{gs} is dominant, then adding more current buys less advantage. The delay for four stages can also be calculated as

$$\begin{aligned}\mathrm{Delay}_n &= 1.5 \cdot RC_{\mathrm{total}} \cdot n \\ &= 1.5 \cdot 5\ \mathrm{k}\Omega(22.2\ \mathrm{fF}) \cdot 4,\ 1.5 \cdot 500\Omega(42\ \mathrm{fF}) \cdot 4,\ 1.5 \cdot 50\Omega(242.2\ \mathrm{fF}) \cdot 4 \\ &= 666\ \mathrm{ps},\ 126\ \mathrm{ps},\ 72.7\ \mathrm{ps}\end{aligned} \tag{5.40}$$

Thus, theoretically, the 100-μA design could function at a maximum speed of approximately 1 GHz. So, for the purposes of exploring this circuit, it will be driven with a 1-GHz, 500-mV, ideal square wave, and the outputs will be observed. The output versus input voltage for the three designs is shown in Figure 5.6. Note that the 100-μA design never fully reaches its output level. This is expected since the time to switch is estimated as 555 ps, but the input switches every 500 ps. Thus, it never fully reaches its maximum output level and slews for the entire 500 ps allowed. The 1-mA design takes 120 ps to rise from about 10% to 90% of its final value, which is in line with prediction. The 10-mA waveform changes from about 10% to 90% of its final value in about 80 ps. This is also reasonably close to the prediction.

Figure 5.7 shows the delay through each of the four stages for the inverters with tail current set to 1 mA. From this diagram, it can be seen that each stage has a delay of about 30 ps (calculated by measuring the time between zero crossings of consecutive stages) for a total of about 120 ps. Again, this is close to the predicted value. The other two delay times can be calculated from Figure 5.6 as 80 ps for

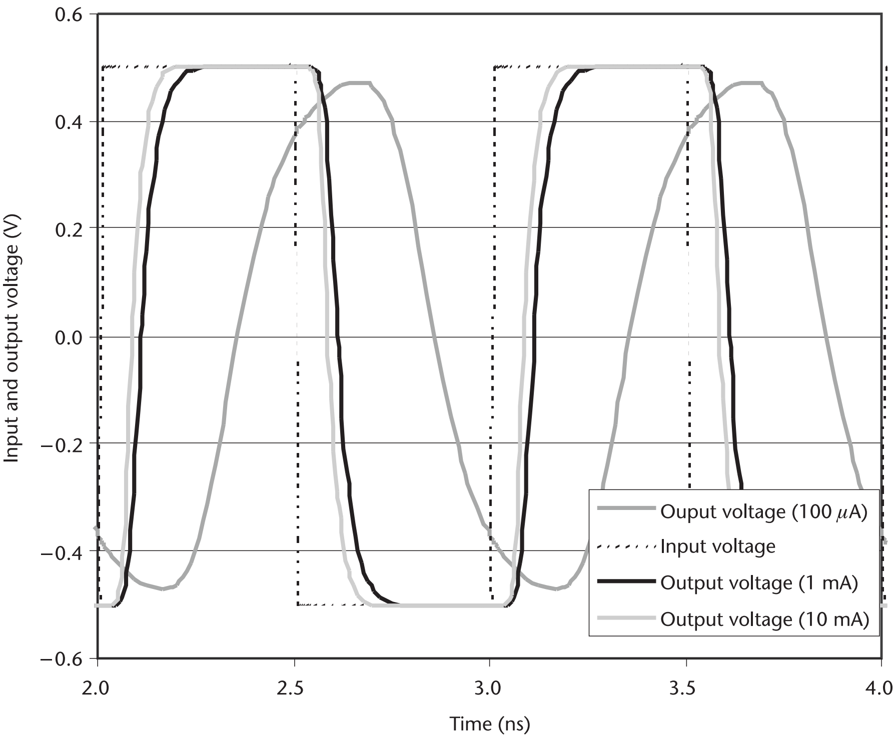


Figure 5.6 Input and output voltages for a set of four CML inverter stages.

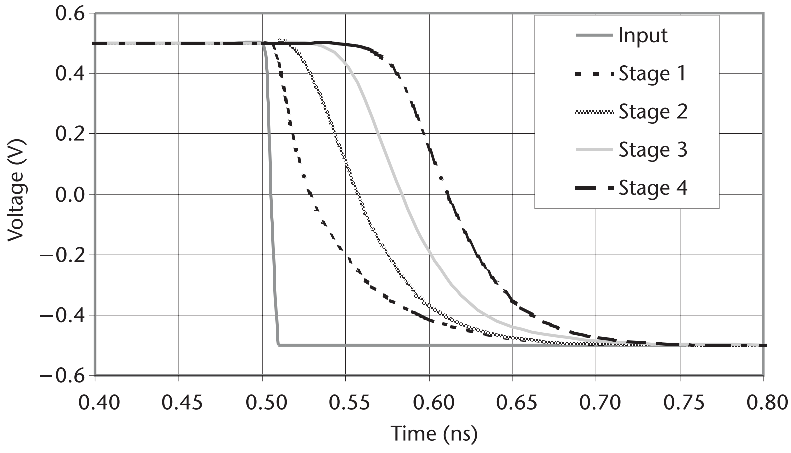


Figure 5.7 Input and output voltages for a set of four CML inverter stages showing the voltage switching in each stage for a 1-mA current setting.

the inverters with 10 mA of current and as 350 ps in the inverters with 100 μA of current. Note that in this last instance, the prediction does not line up well with the simulation results because the waveform shapes are no longer close to the shapes assumed in the derivation.

Drain current in one transistor in the last stage can be compared for the three current levels and is shown in Figure 5.8. Note that the drain current appears more like a square wave than does the output voltage, even for the 100-μA current level. Note that each curve has been normalized for easy comparison.

5.6 CML Combinational Circuits

For high-speed applications and for low switching noise, synthesizers do not always use standard CMOS logic, but use CML instead. By choosing the polarity of the outputs appropriately, the differential pair already discussed can be used to make an inverter. Other common logic functions, such as AND, OR, and XOR gates, can be implemented as shown in Figure 5.9. If the inputs A and B are assumed to be square waves of sufficient amplitude to switch the current flowing through the transistors, then it is easy to prove correct functionality by tracing out the current flow. For instance, in the OR gate, if A is high, then the current must flow through M_2 and M_5 (note that M_5 is included for level matching), pulling the negative output low and, thus, producing logic one. If A is low, then B determines the output. With the differential CML topology, the CML AND gate is exactly the same as the CML OR gate in structure, but the input and output polarities are changed. This is not surprising, as DeMorgan's law points out that

$$\overline{A \cdot B} = \overline{A} + \overline{B} \tag{5.41}$$

The XOR gate looks like a mixer, not surprisingly, since it is the same circuit except that, in a mixer, A would be small enough so that M_1 and M_2 would not switch but would perform as a linear amplifier. As an XOR gate, it can be shown that if either A or B, but not both, is high, a logic one is the result.

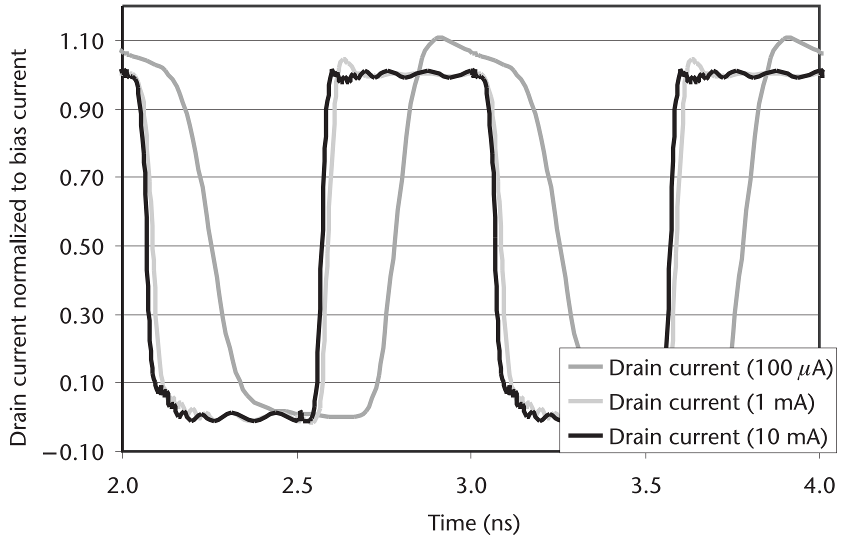


Figure 5.8 Drain current of one transistor in the last stage of a set of four CML inverter stages.

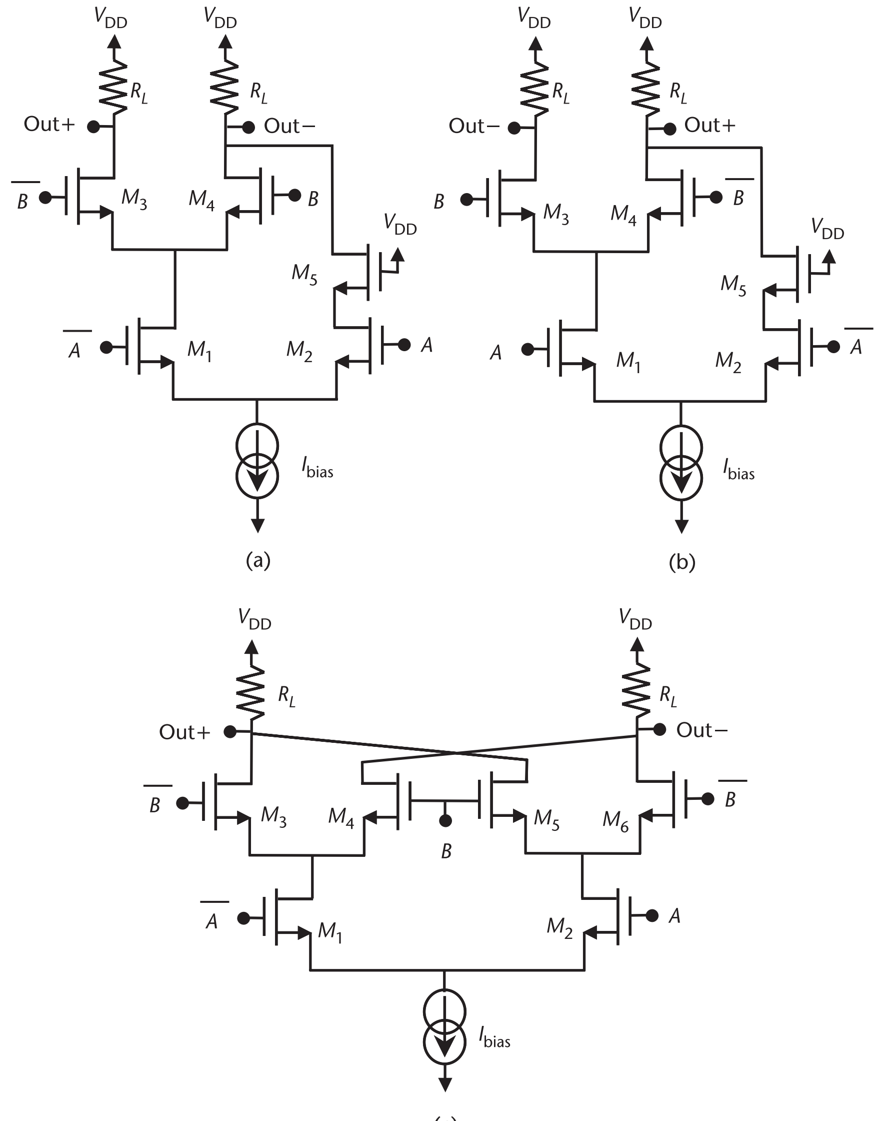


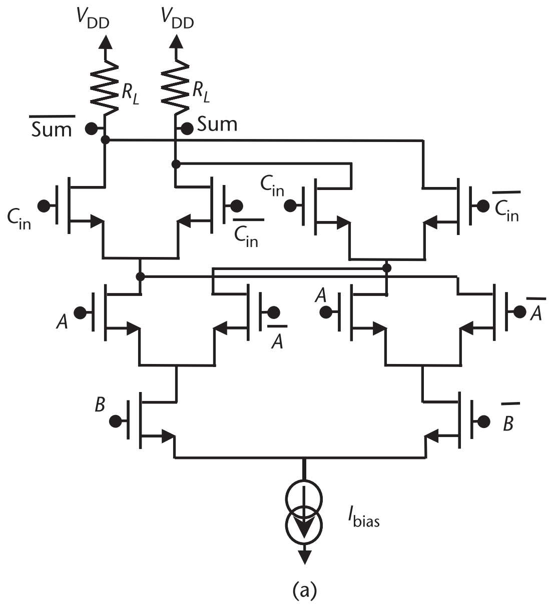
Figure 5.9 Simple CML logic gates: (a) an OR gate, (b) an AND gate, and (c) an XOR gate.

An adder is another commonly used combinational logic block. Figure 5.10 illustrates the CML logic circuits for the sum and carry-out of a 1-bit full adder, which implements the following logic expressions:

$$\text{Sum} = A \oplus B \oplus C_{\text{in}} \tag{5.42}$$

$$C_{\text{out}} = A \cdot B + A \cdot C_{\text{in}} + B \cdot C_{\text{in}}$$

where C_{in} is the carry-in of the full adder. When a ripple adder is formed using a half adder for the least significant bit (LSB) and cascaded, 1-bit full adders for all other bits, the carry-in of the full adder is connected to the carry-out of the previous



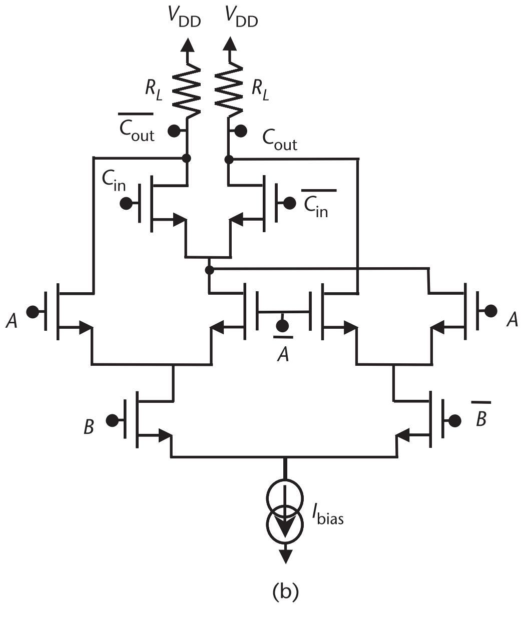


Figure 5.10 A CML full adder logic circuit: (a) sum circuit, sum = $A \oplus B \oplus C$, and (b) carry-out circuit, $C_{out} = A \cdot B + A \cdot C_{in} + B \cdot C_{in}$.

bit, as illustrated in Figure 5.11. Since the carry-in bit arrives later than the input bits A and B, C_{in} is connected to the upper level of the CML circuits, where the delay to the output is minimum.

5.7 CML Sequential Circuits

Unlike combinational logic circuits, sequential logic circuits have memory functions due to built-in feedback. Latches and flip-flops are building blocks of sequential

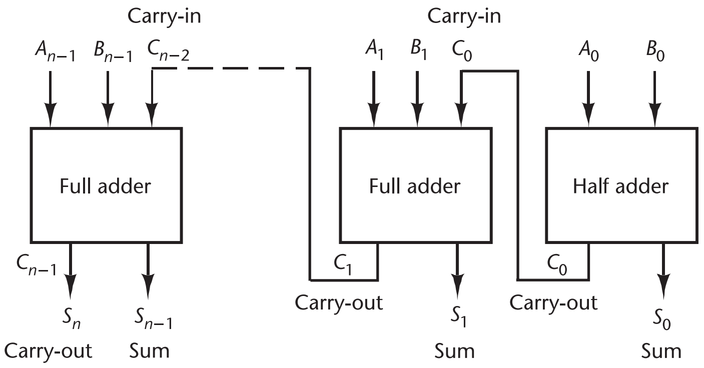


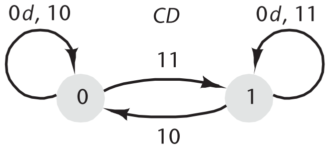
Figure 5.11 A ripple adder formed by a half adder and cascaded full adders.

logic circuits. Although both latches and flip-flops are memory devices, a latch differs from a flip-flop in that a latch is not edge-triggered, with the result that its output follows the input variations during the clock active phase, while a flip-flop is edge-triggered such that its output updates only at the clock transition. As will be shown, two latches can be used to make a master-slave flip-flop, which is ultimately used for the design of CML sequential circuits, such as dividers and phase detectors. A gated delay latch (D-latch) has two inputs, a clock CLK used as the enable signal and an input signal D. When the clock goes high, the latch holds the previous value of the input until the clock goes low again. As shown in Figure 5.12, a D-latch operates as follows: when the clock is high, the latch is enabled and operates in "update" mode; that is, its output Q follows the input signal D. When the clock is low, the latch is in "hold" mode; that is, its output Q holds the previous value. The D-latch state diagram is shown in Figure 5.12(b), where the two-digit number CD denotes the clock input and the data input, respectively, and d represents the "don't care" input. A symbol for an active-high clocked D-latch is shown in Figure 5.12(c). Note that the circle in the output line indicates inverted signal polarity.

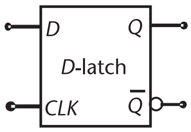
The above-discussed D-latch characteristics can be expressed using the following logic equation:

Enable input CLK	Excitation input D	Present state Q	Next state $Q+$	
0	x	0	0	Hold
0	x	1	1	
1	0	0	0	Update 0
1	0	1	0	
1	1	0	1	Update 1
1	1	1	1	

(a)



(b)



(c)

Figure 5.12 Clocked D-latch (a) excitation table, (b) state diagram, and (c) symbol.

$$Q^+ = D \cdot \mathrm{CLK} + \overline{\mathrm{CLK}} \cdot Q \tag{5.43}$$

where Q^+ indicates the next state value of the latch output. Figure 5.13 illustrates a typical timing diagram for a positive-level sensitive D-latch. As shown, the output of the latch follows the input change during the clock positive phase. Hence, the latch is not an edge-triggered device. Instead, the clock signal is an enable signal; for this reason, the latch is also called a level-triggered memory device to differentiate it from the edge-triggered flip-flops to be discussed in Section 5.8.

A MOS CML latch is usually implemented as shown in Figure 5.14. A bipolar latch has the same topology. When *CLK* is low, all current is passed through M_1, and M_2 is off. Thus, M_5 and M_6 are also off and do nothing. In this state, the latch behaves as if it were a differential pair, and the output follows the input. When *CLK* goes high, M_1 turns off, turning off M_3 and M_4. In this state, M_5 and M_6 turn on. These two transistors are connected in positive feedback, which latches the output value.

To understand this, consider the case when *CLK* goes high, and Q is also high. In this case, the gate of M_6 is at V_{DD}, while the gate of M_5 is low. Thus, current through M_2 will be drawn mostly by M_6 because it will have a much higher V_{GS} than does M_5. Since little current is drawn through M_5, the high value of Q will be reinforced, and since M_6 is drawing current, the low value of $\overline{Q}$ will be reinforced. If Q had been low, M_5 would instead have turned on, reinforcing the low value of Q. Thus, with M_2 drawing current, the latch has been shown to be "holding" its state.

Many variants of this basic circuit are possible. One variant useful for implementing a divide-by-three with 50% duty cycle is a latch with an invertible clock [4], shown in Figure 5.15(a). Here, when the θ input is high, the clock signal passes through transistors M_7 and M_{10}, but when θ is low, the clock signal is inverted, passing through M_8 and M_9.

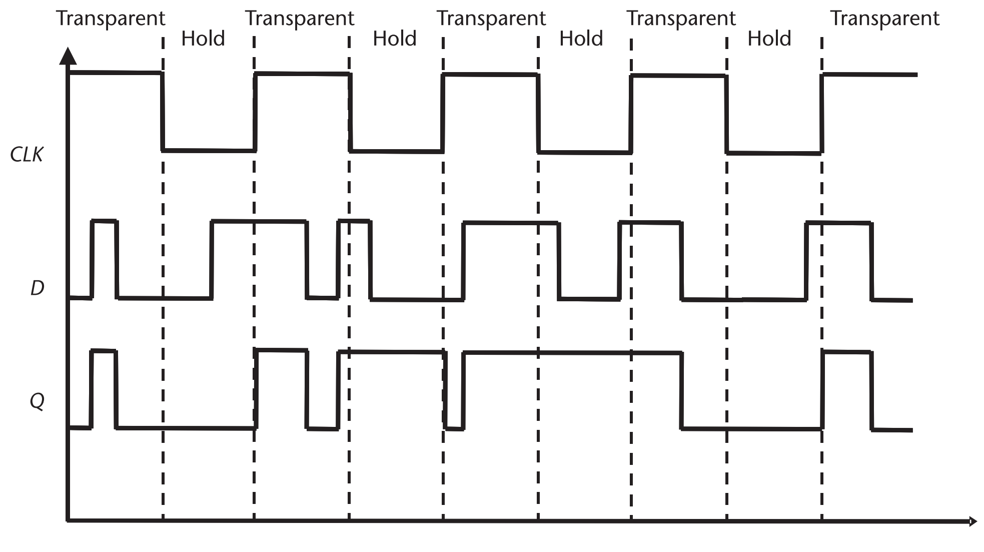


Figure 5.13 Timing diagram showing a typical D-latch operation.

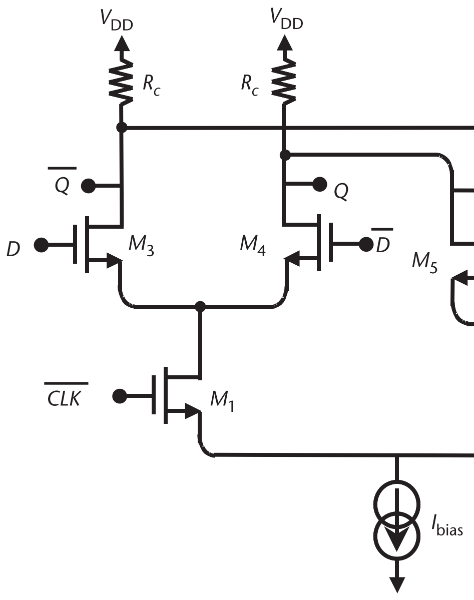


Figure 5.14 CML *D*-latch circuit.

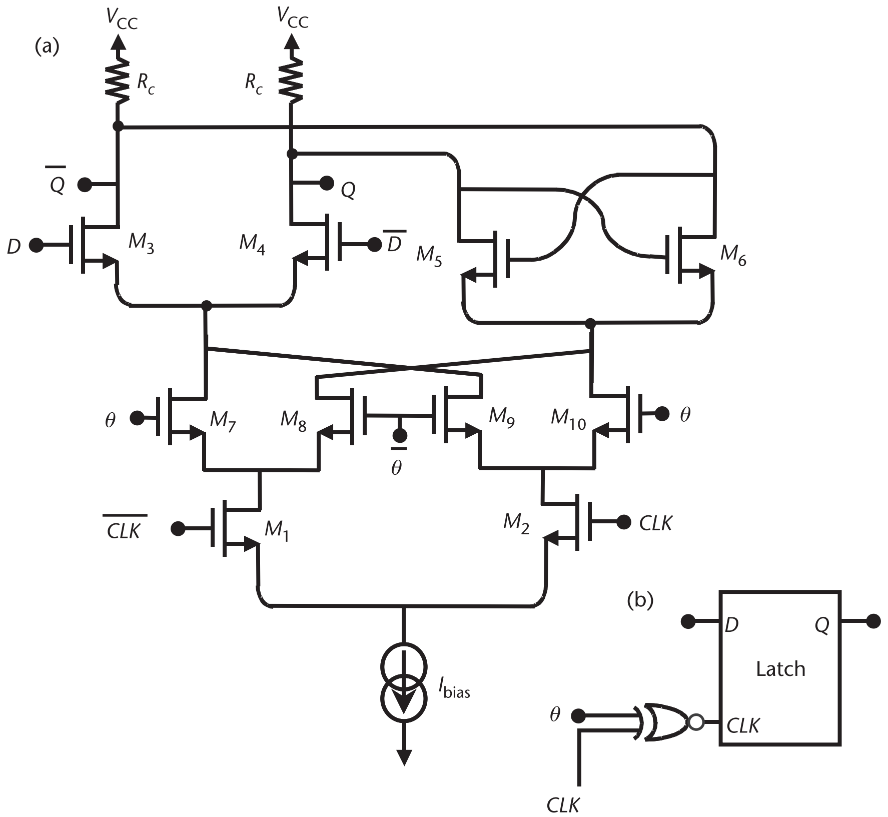


Figure 5.15 CML latch with invertible clock: (a) circuit implementation, and (b) logic diagram.

In comparison to the simple latch circuit shown in Figure 5.14, the extra transistors M_7 through M_{10} act like an XOR, gating the clock. Thus, the logic diagram for this latch needs to be modified to include this, as shown in Figure 5.15(b).

The major advantage to adding the XOR gate, as shown in Figure 5.15, is that stacking the circuits reduces the total current. This can only happen if the supply voltage in the process is high enough to allow such stacking. If such stacking is not possible, the functions can still be implemented, but more tail-current sources will be needed to achieve the same logic function. Stacking can also be done with the input as well as the clock. For example, Figure 5.16 shows an input that incorporates an AND gate and a latch into one tail current.

Often it is desirable to be able to reset the latch to a known state, then to release the reset asynchronously so that all the latches clocked by the same signal can start their operations simultaneously. This is particularly important in order to synchronize the flip-flops built using latches. If the supply voltage is high enough to allow four-level transistor stacking, a resetable latch can be implemented by inserting a pair of reset transistors (Q_7 and Q_8) below the clock and input transistors, as shown in Figure 5.17. Note that $RST\_$ is a convention for active-low signals. When the reset signal $RST\_$ is low, transistor Q_8 is turned on, which forces the current flow through the positive output path and causes the output Q to be

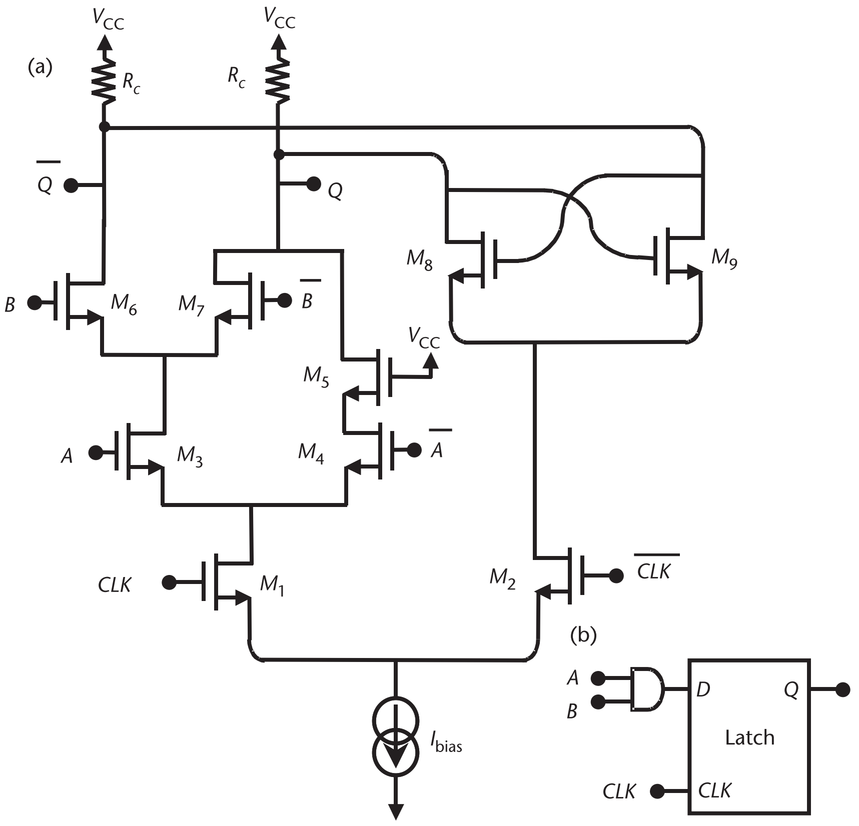


Figure 5.16 CML latch stacked with an AND gate: (a) circuit implementation, and (b) logic diagram.

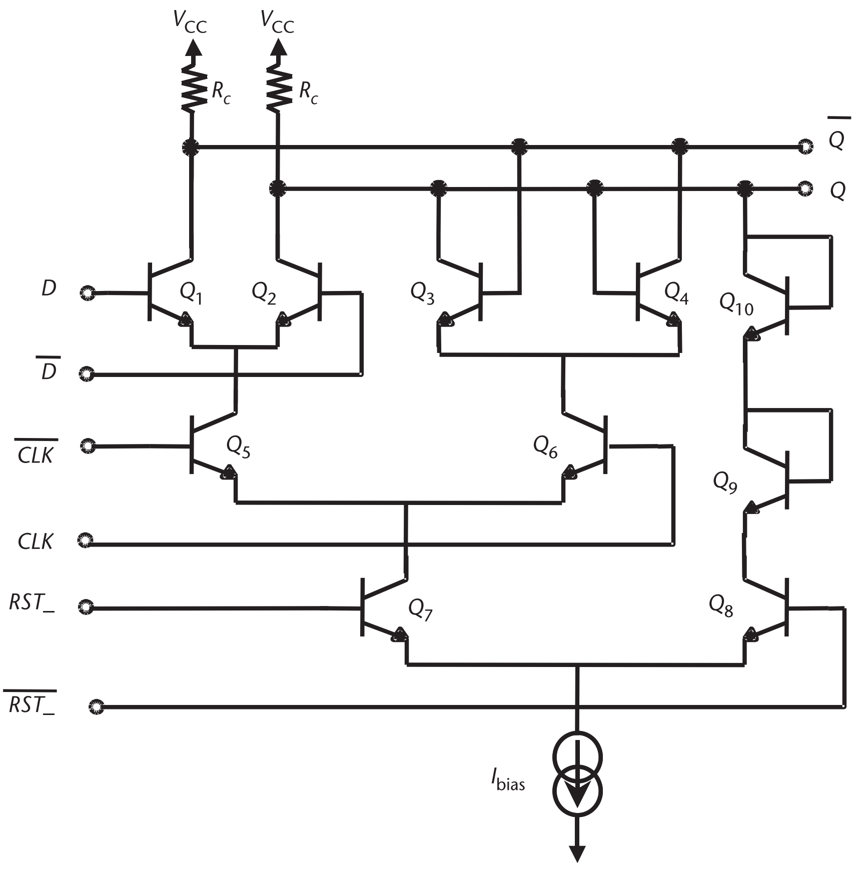


Figure 5.17 CML latch with active-low reset using four-level transistors.

low. Meanwhile, transistor Q_7 is turned off, which causes the output $\overline{Q}$ to be high. Overall, the latch is reset to zero when *RST_* is low. However, with supply voltages lower than about 2.4V, it is very hard to keep all the bipolar transistors from saturation in four-level CML logic. A three-level, resetable latch can be implemented by placing the reset transistors in parallel with the latch transistors, as shown in Figure 5.18 [5]. The structure is a latch followed by an AND gate that provides the reset function, with the final outputs fed back to clear the latch's internal state during the reset mode. This topology is very suitable for low supply voltage applications.

5.8 Master-Slave *D*-Flip-Flop

As shown, the output of a latch is not stable. During the update phase, any input change will be passed to its output. Moreover, the transition of the latch output is not synchronized to the clock edge. The problem can be solved by cascading two latches in a master-slave (MS) configuration, as shown in Figure 5.19. The two latches are driven by complementary clocks such that only one latch is active at a time. During the negative phase of the clock, the master latch is active, and

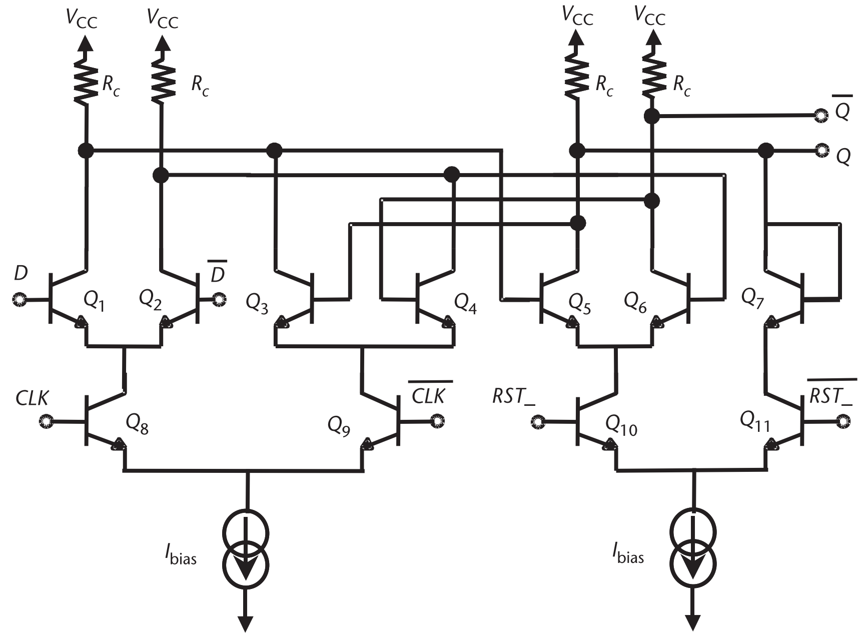


Figure 5.18 CML latch with active-low reset using three-level transistors for low-supply voltage applications.

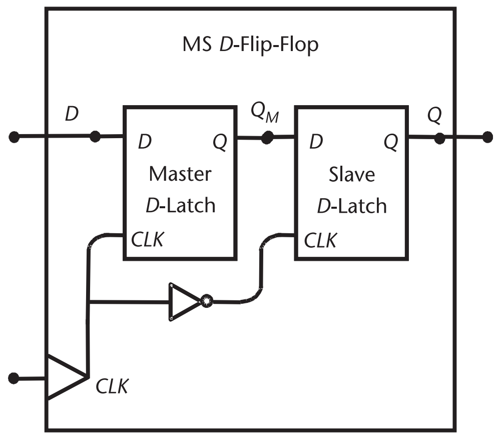


Figure 5.19 MS *D*-flip-flop using two *D*-latches.

its output Q_M is updated according to current input, while the slave latch is disabled, preventing the flip-flop output Q from changing. At the rising edge of the clock, the master latch changes from update to hold mode, while the slave latch becomes enabled, passing the last captured Q_M value to the flip-flop output Q. During the positive phase of the clock, the master latch is disabled, preventing the flip-flop output Q from changing, even if the slave latch is in update mode. Thus, the transition of the MS flip-flop output is only allowed at the rising edge of the clock signal. From this point of view, the MS flip-flop is an edge-triggered device. However, data captured in an MS flip-flop is not quite edge-triggered. Any input change during the negative phase of the clock will be captured at the master latch output Q_M, although only the last update captured right before the clock rising edge will be passed to the flip-flop output.

The behavior of the MS D-flip-flop can be summarized as

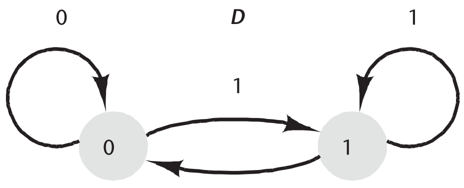
$$Q^+ = D \tag{5.44}$$

and is also shown in Figure 5.20.

Therefore, the new output (Q^+) of the D-flip-flop always assumes the value of its input (D) at the clock rising edge. Note that there is a delay through the flip-flop. Thus, if a signal arrives at the input of the flip-flop, it must wait one clock cycle to be passed to the output. Thus, a flip-flop is one way to implement a unit delay; in other words, it has a transfer function of z^{-1}. The MS D-flip-flop can be implemented in a CML topology as shown in Figure 5.21. Note that emitter followers (labeled EF in Figure 5.21) are used in series with the latch cross-coupling transistors to prevent the latch transistors from saturating and to provide drive

D	Q	C	Q^+
0	0	⎍	0
0	1	⎍	0
1	0	⎍	1
1	1	⎍	1

(a)



(b)

Figure 5.20 MS *D*-flip-flop: (a) excitation table, and (b) state diagram.

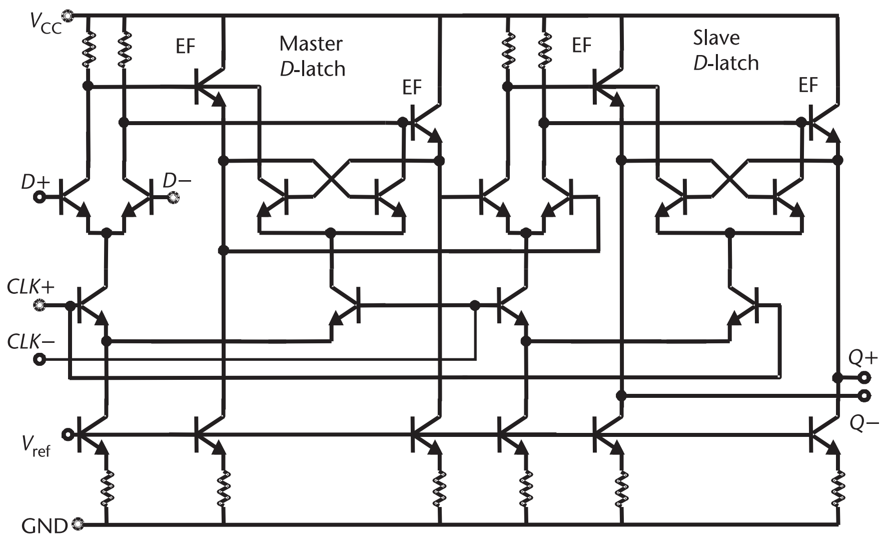


Figure 5.21 CML MS *D*-flip-flop.

capability between the stages. If power consumption is a concern or the supply voltage does not provide enough headroom for a four-level transistor topology, the emitter followers can be omitted. In such a case, the latch transistors are soft-saturated with an ac swing larger than 200 mV since their base and collector are biased at the same dc voltage.

5.9 CML Circuit-Delay Analysis

To understand the CML circuit operation speed, the delay-time constants need to be modeled for a series-gated bipolar CML D-latch, which is the common basic building block of many high-speed switching circuits. Figure 5.22 illustrates the D-latch, half-circuit, small-signal model, where R_{c_pre} denotes the load resistance of the previous CML stage. For worst-case propagation delay, the upper-level data inputs are set as constant, and a step is provided at the input of the lower-level clock transistors [6, 7]. We assume that transistors Q_3 and Q_5 are on, and transistors Q_4 and Q_6 are off. The load parasitic capacitance C_{L2} is ignored since only dc current flow occurs in the half-circuit analysis. The analysis of the half circuit is sufficient for differential operations. The delay model can apply not only to CML sequential circuits, such as the D-latch, but also to any combinational circuits, such as AND, OR, and XOR gates.

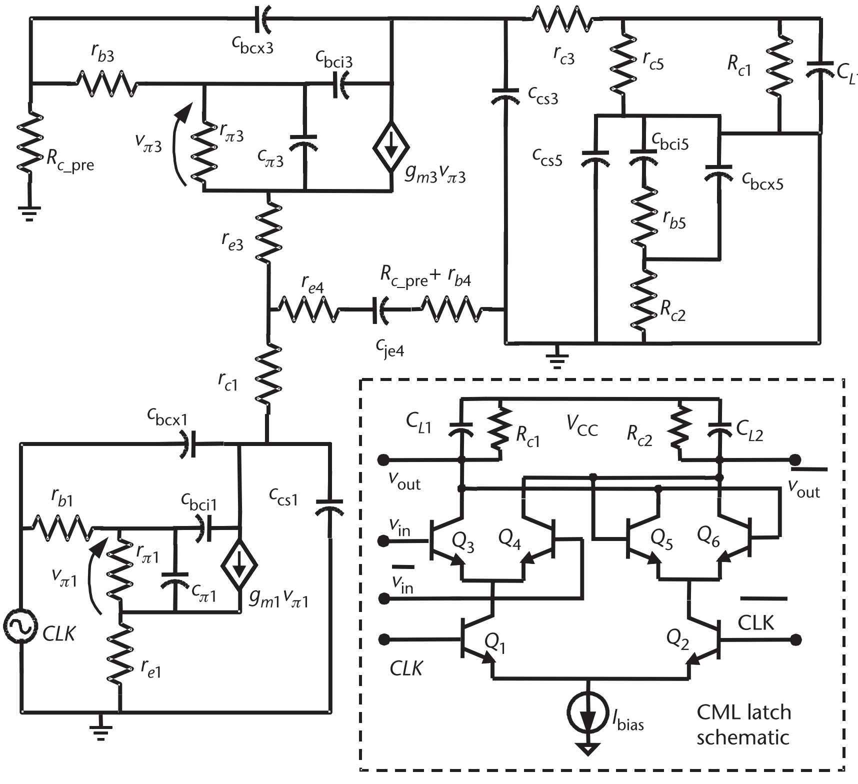


Figure 5.22 D-latch, half-circuit, small-signal model.

The delay through the CML D-latch can be expressed as the sum of RC time constants, assuming dominant pole behavior [8]. The time constants associated with various capacitors for lower transistors in the equivalent circuit model are given by

$$\begin{aligned}
\tau_{\pi 1} &= \frac{r_{e1} + r_{b1}}{1 + g_{m1} r_{e1} + \dfrac{r_{e1} + r_{b1}}{r_{\pi 1}}} C_{\pi 1} \\
\tau_{bcx1} &= \left(r_{c1} + r_{e3} + \frac{r_{\pi 3} + r_{b3} + R_{c\_pre}}{\beta + 1} \right) C_{bcx1} \\
\tau_{bci1} &= \left\{ \left(r_{c3} + r_{e3} + \frac{r_{\pi 3} + r_{b3} + R_{c\_pre}}{\beta + 1} \right) \right. \\
&\quad \left. + r_{b1} \left[1 + \frac{g_{m1} \left(r_{c1} + r_{e3} + \dfrac{r_{\pi 3} + r_{b3} + R_{c\_pre}}{\beta + 1} + r_{e1} \right)}{1 + g_{m1} r_{e1}} \right] \right\} \cdot C_{bci1} \\
\tau_{cs1} &= \left(r_{c1} + r_{e3} + \frac{r_{\pi 3} + r_{b3} + R_{c\_pre}}{\beta + 1} \right) C_{cs1}
\end{aligned} \tag{5.45}$$

The time constants for upper transistors are listed as follows:

$$\begin{aligned}
\tau_{\pi 3} &= \frac{1}{\left(g_{m3} + \dfrac{1}{r_{\pi 3}} \right)} \cdot C_{\pi 3} \\
\tau_{bci3} &= (2R_{c\_pre} + r_{c3} + r_{b3}) \cdot C_{bci3} \\
\tau_{bcx3} &= (2R_{c\_pre} + r_{c3}) \cdot C_{bcx3} \\
\tau_{bci5} &= (2R_{c\_pre} + r_{c5} + r_{b5}) \cdot C_{bci5} \\
\tau_{bcx5} &= (2R_{c\_pre} + r_{c5}) \cdot C_{bcx5}, \ \tau_{cs3} = (R_{c\_pre} + r_{c3}) \cdot C_{cs3} \\
\tau_{cs5} &= (R_{c\_pre} + r_{c5}) \cdot C_{cs5} \\
\tau_{je4} &= (R_{c\_pre} + r_{b4} + 2r_{e4}) \cdot C_{je4} \\
\tau_{cload} &= R_{C1} \cdot C_{L1}
\end{aligned} \tag{5.46}$$

where subscripts c, b, e, and s mean collector, base, emitter, and substrate of the corresponding transistors; π denotes base-emitter junction; subscripts i and x denote the intrinsic and extrinsic parts of the base-collector capacitances; and subscript je denotes the junction capacitance. The delays associated with upper and lower transistors can be found by summing all the delays in (5.45) and (5.46):

$$t_{\text{lower}} = \tau_{\pi 1} + \tau_{\text{bci1}} + \tau_{\text{bc1}} + \tau_{\text{cs1}}$$
$$t_{\text{upper}} = \tau_{\pi 3} + \tau_{\text{bci3}} + \tau_{\text{bcx3}} + \tau_{\text{bci5}} + \tau_{\text{bcx5}} + \tau_{\text{cs3}} + \tau_{\text{cs5}} + \tau_{\text{je4}} + \tau_{\text{cload}} \quad (5.47)$$
$$t_{\text{total}} = 0.69 \times (t_{\text{lower}} + t_{\text{upper}})$$

At any junction where voltages change rapidly by a large amount, those junction capacitances are modified from zero bias value C_{jo} as $K \cdot C_{jo}$ by using coefficient K, given by [6]

$$K = \left(\frac{\phi^m}{V_2 - V_1}\right)\left[\frac{(\phi - V_1)^{1-m}}{1-m} - \frac{(\phi - V_2)^{1-m}}{1-m}\right] \quad (5.48)$$

where the ac swing is from voltage V_1 to V_2, and an ac voltage swing of ±150 mV_{pp} is considered to be large, ϕ is the built-in potential across the junction under zero bias, and m is the grading coefficient and equals 1/2 for an abrupt junction. The transistor capacitances are assumed to be constant when bias current varies, which is a good approximation at low bias current.

5.10 Low-Power CML Circuits

CML circuit design always involves trade-offs between power consumption and speed. The delay model developed in the previous sections can be used to optimize CML circuits to improve circuit performance in terms of power consumption and speed. Using the delay models developed in previous sections, Figure 5.23 shows the CML D-latch propagation delay with respect to the bias current [9].

It comes as no surprise that the optimum biasing current for minimum delay is the transistor peak f_T current. According to Figure 5.23, it is obvious that there is not much speed improvement by increasing the biasing current beyond 60% of the peak f_T current. Biasing the circuit close to the peak f_T current may cause the actual bias current to go beyond the peak f_T current under temperature, supply, and process variations, which leads to a dramatic speed penalty as a result of current crowding and conductivity-modulation effects in the base region. Unless the absolute maximum speed of operation is required, it is good practice to bias the CML circuit with less than 60% of the peak f_T current to save unnecessary power consumption. Figure 5.23 shows that biasing the CML circuits at about 60% of the peak f_T current (0.9 mA) can achieve about 80% of the maximum speed that would have been achieved at the peak f_T current.

Moreover, it is evident that the CML latch delay is dominated by the delay associated with the upper transistors. Hence, reducing the delay due to upper-level transistors is critical to improving CML switching speed. The optimum bias current for minimum delay is not the same for upper- and lower-level transistors. The lower transistors have minimum delay at lower bias current. It is thus intuitive that there will be a speed improvement if the CML circuit is biased with slightly higher bias current for the upper-level transistors than for the lower-level transistors. For instance, with the same total bias current, the bias currents can be reduced by about 20% for the lower-level transistors and increased by 20% for the upper-

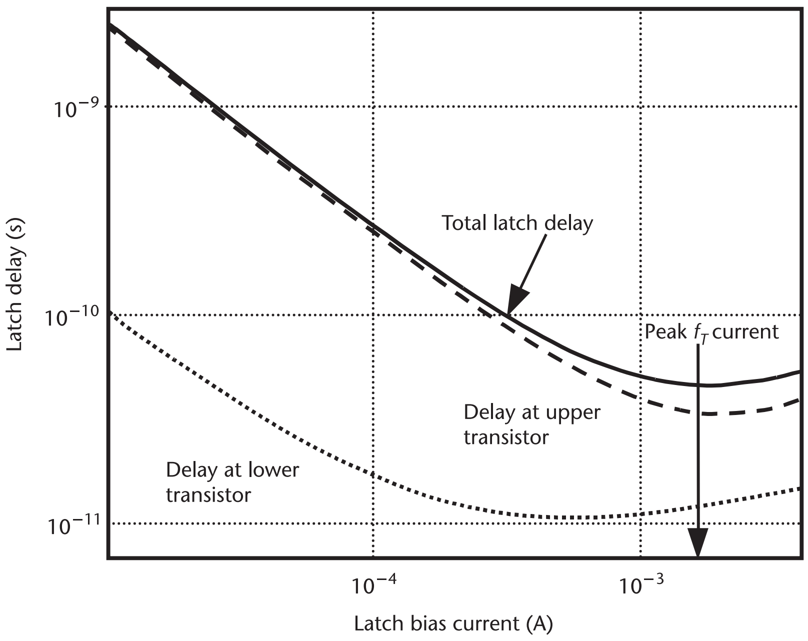


Figure 5.23 CML D-latch delay versus bias current for a transistor size of 0.5 μm by 2.5 μm.

level transistors. Figure 5.24 illustrates a modified CML D-latch biased in such a manner. The technique is called "keep alive" since there will always be a small amount of bias current flowing through the upper-level transistors, keeping them alive in slightly on states, regardless of the clock and data. As a result, the capacitors associated with the upper-level transistors, which are the dominant contributors

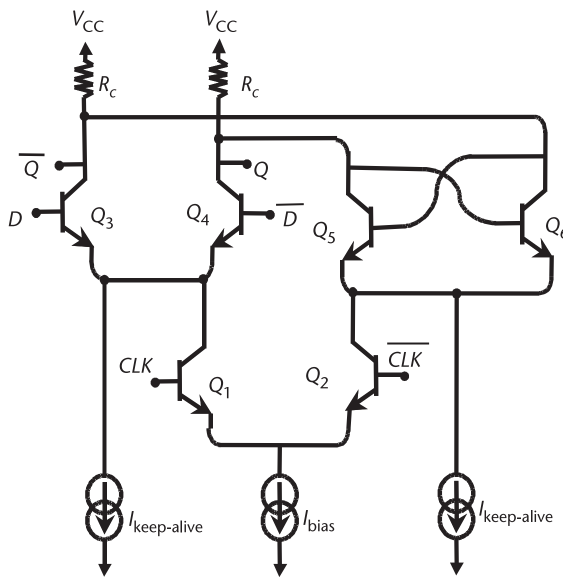


Figure 5.24 CML D-latch with keep-alive topology.

to the CML propagation delay, will be precharged to a certain level before the clock is enabled. When the clock is enabled, the upper-level capacitors will need relatively less time to reach their steady-state values. Moreover, optimization can also be performed in terms of transistor sizing for upper and lower transistors. A speed improvement of about 11% can be achieved by using the keep-alive CML topology [9]. The keep-alive technique does not increase the power consumption, but the output noise margin (voltage swing) is slightly reduced; that is, noise margin is traded for circuit speed and power consumption.

5.11 CML Biasing Circuits

So far, all the current sources in the CML circuits have been shown as ideal, but these are an important part of the design as well. Their design depends on the technology being used, but, fundamentally, a current source consists of a current mirror of some kind, with the reference current typically generated by a bandgap circuit.

In bipolar technology, a current-mirror biasing circuit could look like that shown in Figure 5.25(a). In this mirror, the current is scaled up N times from the reference produced by the bandgap circuit. Resistors R_E and NR_E are included to increase the output impedance of the current mirror and to improve matching between the diode connected transistor Q_1 and the current source transistor NQ_1. This is because the current-voltage relationship in a bipolar transistor is exponential; thus, the sensitivity of the output current to changes in V_{BE} is greater than the sensitivity of the output current to changes in R_E. An additional feature of this current source is the addition of transistor Q_2 to provide base current to the main mirror transistors, thus improving current matching. The capacitor C is included to reduce the circuit noise, and R_{BL} is included to ensure that transistor Q_2 is always biased in the active region.

The output impedance of the current mirror can be computed with the aid of Figure 5.25(b). Here, it is assumed that transistor Q_1, being essentially diode

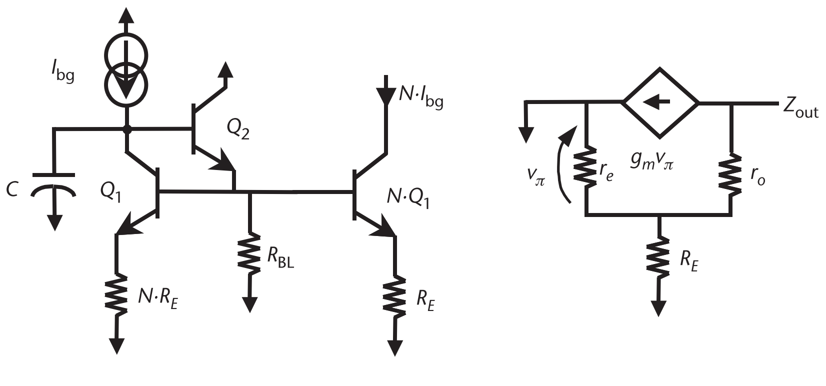


Figure 5.25 (a) A bipolar current source, and (b) a simplified model for calculating the output impedance.

connected, provides a short circuit (low impedance path) to ground. Then, the output impedance of this circuit is given by

$$Z_{\text{out}} = \frac{R_E//r_e + r_o}{1 - g_m R_E//r_e} \tag{5.49}$$

Note that, as expected, the value of Z_{out} reduces to be simply r_o if no degeneration is added to the circuit; however, it is clear that even a small amount of degeneration will add significantly to the output impedance of the circuit. This impedance, when combined with parasitic capacitance connected to the output of the current mirror, can be used to determine at which frequency the output impedance will start to decrease and, thus, help to determine the frequencies at which this structure will provide a useful current source.

In CMOS technologies, it is not as common to use degeneration resistors as it is in bipolar technology. However, since CMOS typically has much lower output impedance, often cascode transistors are included, as shown in Figure 5.26(a). Also, if current matching is a concern, then the voltage at the drain of both M_3 and M_2 should be matched as closely as possible. These cascode transistors also provide an additional degree of freedom in the design of the mirror. As an example, the switching speed of a CML stage can be affected by the output capacitance of its current source (switching the stage means that second harmonic voltage will appear on the tail-current source). For a single-transistor current source, there is a trade-off between output impedance and capacitance. However, with a cascode current source, the cascode transistor can be chosen to be of a minimum length to minimize capacitance, and longer channel devices can be used for transistors M_1 and NM_1 for high output impedance.

The output impedance of the current mirror can be computed with the aid of Figure 5.26(b). Here, it is assumed that transistor M_1, being essentially diode connected, provides a short circuit (low impedance path) to ground. This means that the current source of M_2 is not active as the gate and source are ac grounded. Thus, the output impedance of this circuit is given by

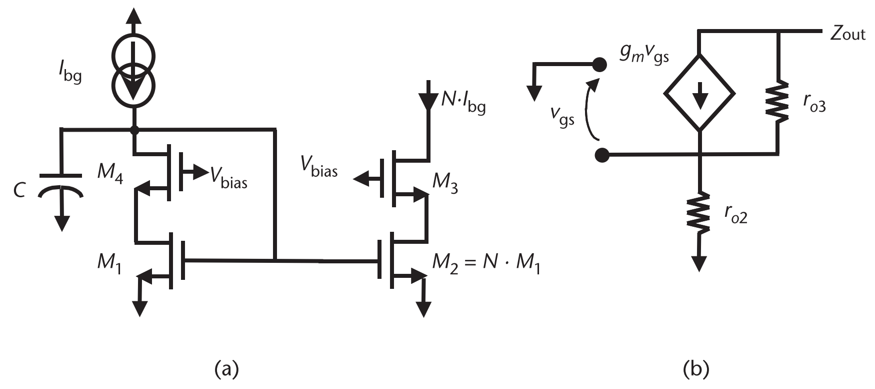


Figure 5.26 (a) A CMOS current source and (b) a simplified model for calculating the output impedance.

$$Z_{\text{out}} = \frac{(r_{o3} + r_{o2} - g_m r_{o2} r_{o3})}{1 - 2g_m r_{o2}} \tag{5.50}$$

Note that the value of Z_{out} reduces to be simply r_{o2} if no cascode is used. This impedance, when combined with parasitic capacitance connected to the output of the current mirror, can be used to determine at which frequency the output impedance will start to decrease and, thus, help to determine the frequencies at which this structure will provide a useful current source.

Now it is also necessary to provide reference currents that are supply and temperature independent [10–13]. Since circuit performance in a silicon process is affected by temperature, supply voltage, and process variations, it is possible to find dependencies that cancel one another. To start, consider the bipolar base emitter voltage characteristic described in Appendix B:

$$V_{\text{BE}} = \frac{kT}{q} \ln \frac{I_C}{I_S} \tag{5.51}$$

This expression seems to show that base-emitter voltage is directly proportional to temperature; however, I_S has a large temperature dependence as well. An expression for V_{BE} as a function of temperature is [14, 15]

$$V_{\text{BE}} = V_{\text{BG}}\left(1 - \frac{T}{T_0}\right) + V_{\text{BE0}}\frac{T}{T_0} + \frac{2.3 \cdot kT}{q} \ln \frac{T_0}{T} + \frac{kT}{q} \ln\left(\frac{I_C}{I_{\text{C0}}}\right) \tag{5.52}$$

where T_0 is the reference temperature, V_{BE0} is the base-emitter voltage at the reference temperature, and V_{BG} is the bandgap voltage of silicon (approximately 1.206V). Even though it may not be immediately obvious from this expression, V_{BE} will actually decrease for a constant collector current with increasing temperature. Thus, if the collector current is assumed to be constant, then the derivative of V_{BE} with respect to temperature is

$$\frac{dV_{\text{BE}}}{dT} = \frac{V_{\text{BE0}} - V_{\text{BG}}}{T_0} + \frac{2.3 \cdot k}{q} \ln \frac{T_0}{T} + \frac{-2.3 \cdot k}{q} \tag{5.53}$$

Note that this expression shows that not a lot can be done to adjust the slope of the temperature dependence.

Next, to cancel this negative temperature/voltage relationship, a voltage that increases with temperature must be found. To do this, assume that two BJTs are biased at different currents. Then, the difference between their base-emitter voltages will be given by

$$\Delta V = V_{\text{BE1}} - V_{\text{BE2}} = \frac{kT}{q} \ln \frac{I_{\text{C1}}}{I_S} - \frac{kT}{q} \ln \frac{I_{\text{C2}}}{I_S} = \frac{kT}{q} \ln \frac{I_{\text{C1}}}{I_{\text{C2}}} \tag{5.54}$$

Since (5.54) does not contain I_S, this relationship is much simpler than (5.52), and this voltage is directly proportional to temperature. In this case, the slope of the temperature dependence is given by

$$\frac{d\Delta V}{dT} = \frac{k}{q} \ln \frac{I_{C1}}{I_{C2}} = \frac{k}{q} \ln m \tag{5.55}$$

Therefore, the slope can be adjusted by changing m, the ratio of I_{C1} to I_{C2}, and, by proper choice, can be made equal in magnitude and opposite in sign to (5.53).

A simple circuit that could be used to generate a bandgap reference is shown in Figure 5.27. In this circuit, the op-amp is used to keep the voltage at the collector of Q_1 and the voltage at the top of the resistor R the same by adjusting the value of the V_{GS} of the identical PMOS transistors. Thus, the voltage V_{ref} is given by

$$V_{ref} = V_{BE1} = V_{BE2} + V_R \tag{5.56}$$

where V_R is the voltage drop across the resistor R.

Therefore, the voltage across the resistor R can also be given by

$$V_R = V_{BE1} - V_{BE2} \tag{5.57}$$

V_R is therefore equal to the difference in the two base-emitter voltage drops. Thus, V_{ref} is given by

$$V_{ref} = V_R + V_{BE2} \tag{5.58}$$

and therefore is made up of two voltages that have opposite temperature dependences and consequently this voltage can be made to be independent of temperature. The current through the circuit is given by

$$I_{ref} = \frac{V_{BE1} - V_{BE2}}{R} \tag{5.59}$$

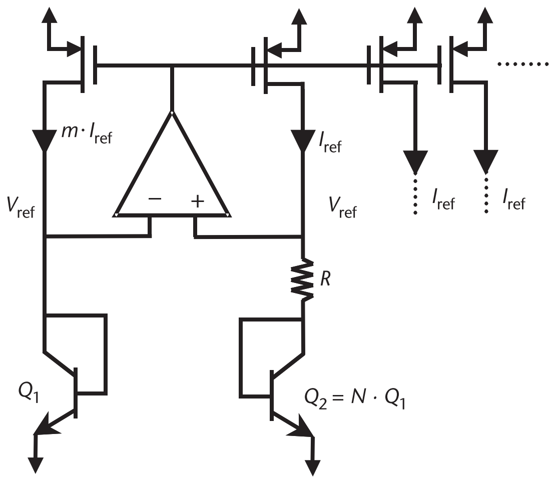


Figure 5.27 A simple bandgap reference generator with an output current proportional to absolute temperature.

Thus, current is proportional to absolute temperature. This current can be scaled or mirrored to other PMOS transistors to create as many copies as is needed.

Note that, in practice, this circuit requires a startup circuit because, in addition to the solution just assumed, the condition where all currents in the circuit are zero will also provide a stable operating point for the op-amp. Startup could be accomplished by injecting some current into the circuit at power up.

5.12 Driver Circuits

In modern ICs, often the synthesizer clock will have to be routed all over the chip. In most ICs, this will mean the need to drive millimeters of interconnect. There is much debate over how best to do this. Regardless of which approach is adopted, the driver circuit will require a large current. As a result, efficient design of these circuits is important. In addition, driving this amount of interconnect will mean that a lot of inductance and capacitance will act to degrade the signal. The two most obvious choices for driver circuits are the inverter and the emitter/source follower circuits. Figure 5.28 shows an emitter follower (using a bipolar transistor). The equivalent circuit with bipolar transistors replaced with MOS transistors would be called a source follower. Emitter and source followers work well at low frequencies, where inductance is not as much of a problem as capacitance, but they tend to work less well at higher frequencies. This is because their low output impedance reduces the effective RC time constant on the line. However, since they drive voltage rather than current, the inductance of the line forms a voltage divider with the load at higher frequencies.

Inverter circuits can also work quite well but often require very large currents to keep their RC time constants low. They can also be connected in a cascode configuration to allow them to drive a fairly low load impedance (although, through switching action, this will not be as low as might be expected), as shown in Figure 5.29.

A superior approach is to combine these two circuits in a push-pull arrangement, as shown in Figure 5.30. Here, a follower circuit is combined with an inverter stage to drive a transmission line. The idea is that for half of the cycle, the follower M_3 supplies current to the transmission line while current is sunk out of the

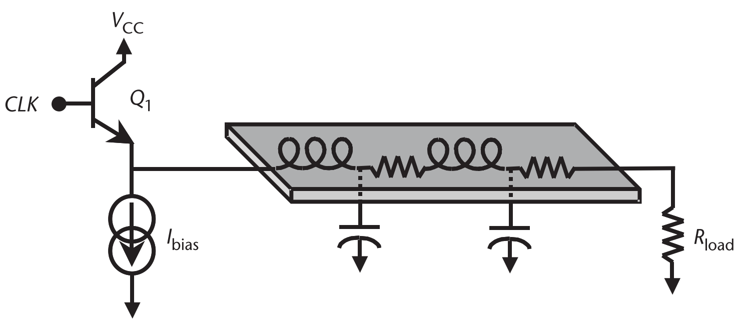


Figure 5.28 Followers as interconnect driver circuits.

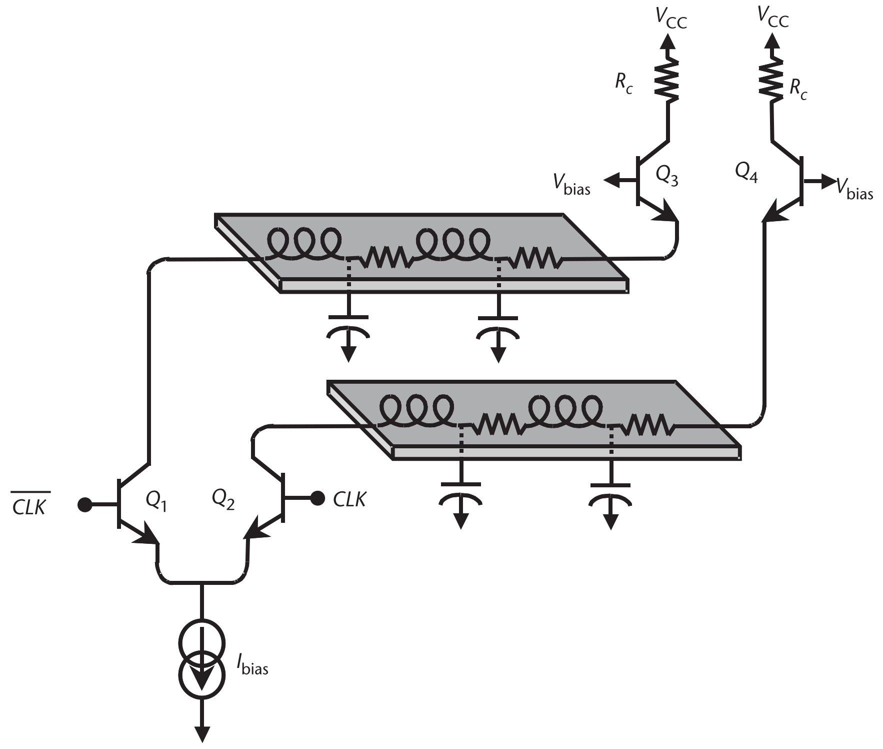


Figure 5.29 Inverter circuits as interconnect driver circuits.

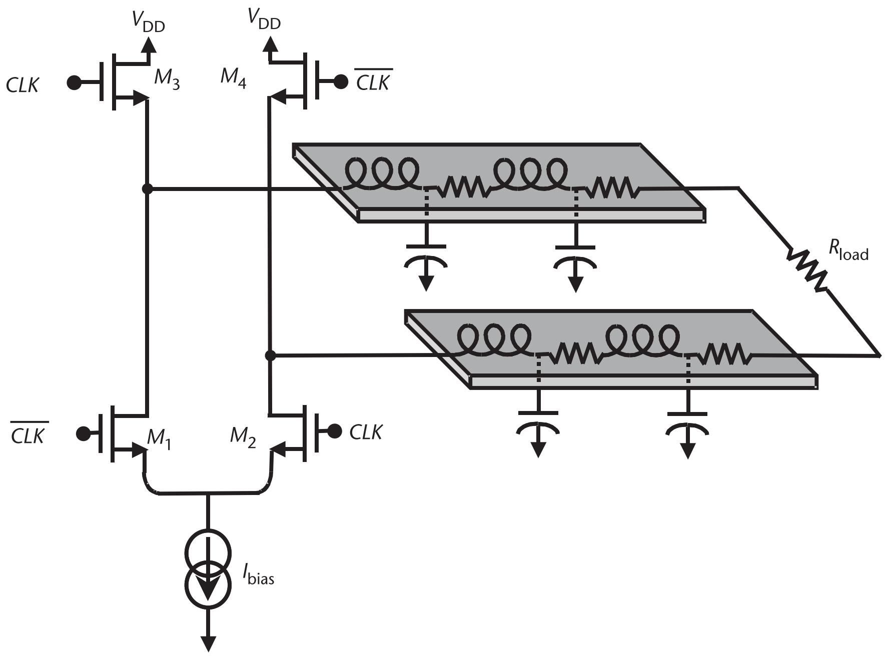


Figure 5.30 Efficient push-pull output buffer.

transmission line by the inverter transistor M_2. Meanwhile, transistors M_4 and M_1 are off. In the other half of the cycle, the follower M_4 turns on, and current is sunk out of the transmission line by the inverter transistor M_1. Since in this circuit almost all the current ends up flowing through the transmission line and very little is wasted just biasing the transistors, this circuit can provide very square-looking waves using minimal current. The only other wrinkle to the proper design of this stage is that, for best operation, the clock edges in the four transistors have to be lined up properly. Since the delay through the two amplifiers is usually different, another stage may have to be used ahead of the actual driver to provide a delay to line up the clock edges.

References

[1] Sedra, A. S., and K. C. Smith, *Microelectronic Circuits*, 5th ed., New York: Oxford Press, 2004.

[2] Jaeger, R. C., and T. N. Blalock, *Microelectronic Circuit Design*, 2nd ed., New York: McGraw-Hill, 2004.

[3] Baker, R. J., H. W. Li, and D. E. Boyce, *CMOS Circuit Design, Layout, and Simulation*, New York: IEEE Press, 1998.

[4] Magoon, R., and A. Molnar, "RF local Oscillator Path for GSM Direct Conversion Transceiver with True 50% Duty Cycle Divide by Three and Active Third Harmonic Cancellation," *IEEE Radio Frequency Integrated Circuit Symposium*, Seattle, WA, 2002, pp. 23–26.

[5] Dai, F. F., et al., "A Low Power 5 GHz Direct Digital Synthesizer Implemented in SiGe Technology," *IEEE 5th Topical Meeting on Silicon Monolithic Integrated Circuits in RF Systems*, Atlanta, GA, September 2004.

[6] Rabaey, J. M., A. Chandrakasan, and B. Nikolic, *Digital Integrated Circuits: A Design Perspective,* 2nd ed., Upper Saddle River, NJ: Prentice Hall, 2003.

[7] Alioto, M., and G. Palumbo, "Modeling and Optimized Design of Current Mode MUX/XOR and D-Flip-Flop," *IEEE Transactions on Circuits and Systems-II*, Vol. 47, No. 5, May 2000, pp. 452–461.

[8] Sharaf, K. M., and M. Elmasry, "An Accurate Analytical Propagation Delay Model for High-Speed CML Bipolar Circuits," *IEEE J. Solid-State Circuits*, Vol. 29, January 1994, pp. 31–45.

[9] Kakani, V., F. F. Dai, and R. C. Jaeger, "Delay Analysis and Optimal Biasing for High Speed Low Power CML Circuits," *IEEE International Symposium on Circuits and Systems (ISCAS)*, Vancouver, Canada, May 2004, pp. 869–872.

[10] Johns, D. A., and K. Martin, *Analog Integrated Circuit Design*, New York: John Wiley & Sons, 1997.

[11] Razavi, B., *Design of Analog CMOS Integrated Circuits*, New York: McGraw-Hill, 2001.

[12] Lee, T. H., *The Design of CMOS Radio Frequency Integrated Circuits*, Cambridge, United Kingdom: Cambridge University Press, 1998.

[13] Grey, P. R., et al., *Analysis and Design of Analog Integrated Circuits*, 4th ed., Cambridge, United Kingdom: Cambridge University Press, 1998.

[14] Brugler, J., "Silicon Transistor Biasing for Linear Collector Current Temperature Dependence," *IEEE J. Solid-State Circuits*, Vol. SC-2, June 1967, pp. 57–58.

[15] Tsividis, Y., "Accurate Analysis of Temperature Effects in I_C–V_{BE} Characteristics with Application to Bandgap Reference Sources," *IEEE J. Solid-State Circuits*, Vol. 15, December 1980, pp. 1076–1084.

CHAPTER 6

Dividers and Phase-Frequency Detectors

6.1 Introduction

This chapter examines the design of dividers and PFDs. The main application for dividers is in the feedback path of PLL-based frequency synthesizers, dividing the oscillator output down to the reference frequency. There are several challenges in divider design, which this chapter will discuss. The oscillator drives the divider input and, since this is the highest frequency in the circuit, speed is a big challenge in divider design. At each subsequent stage, the speed is lower, and the challenge becomes to operate at the required speed at the lowest power dissipation. Another challenge is that dividers need to be programmable and, in many cases, adjustable in real time. They may be dynamically switched between two or more different divider ratios to generate the equivalent of a fractional-divider ratio. Such dividers are called dual-modulus or multimodulus dividers. As a related topic to frequency division, the design of frequency multipliers is also discussed briefly.

The phase detector or PFD directly follows the divider. The PFD compares the divided-down signal with the reference signal and provides an error voltage, which is ultimately fed back to control the oscillator. In this chapter, a variety of design issues and challenges will be discussed. This will include techniques to avoid the dead zone, a situation in which the PFD is not able to respond to the phase difference between the divider output and the reference signal. Another topic for discussion is circuitry to detect the lock condition. Finally, some of the differences in design considerations for phase detectors as used in clock-and-data-recovery (CDR) circuits are also discussed.

6.2 Dividers

Most of the circuits studied in this section will be made with logic gates and latches (not flip-flops). Generally, there will be two sets of latches in each circuit, as shown in Figure 6.1. A clock will drive one set (the *P* set in Figure 6.1), and the other set (the *Q* set in Figure 6.1) will be driven by the opposite phase of the clock. For the purposes of this chapter, the clock will usually be the signal to be divided. Each set of latches will be separated from the other set of latches by logic. It is important to remember that latches have three possible states rather than just two. These states are one, zero, and transparent. In digital circuits made with latches rather than flip-flops, the state of the system is determined by the state of the latches that

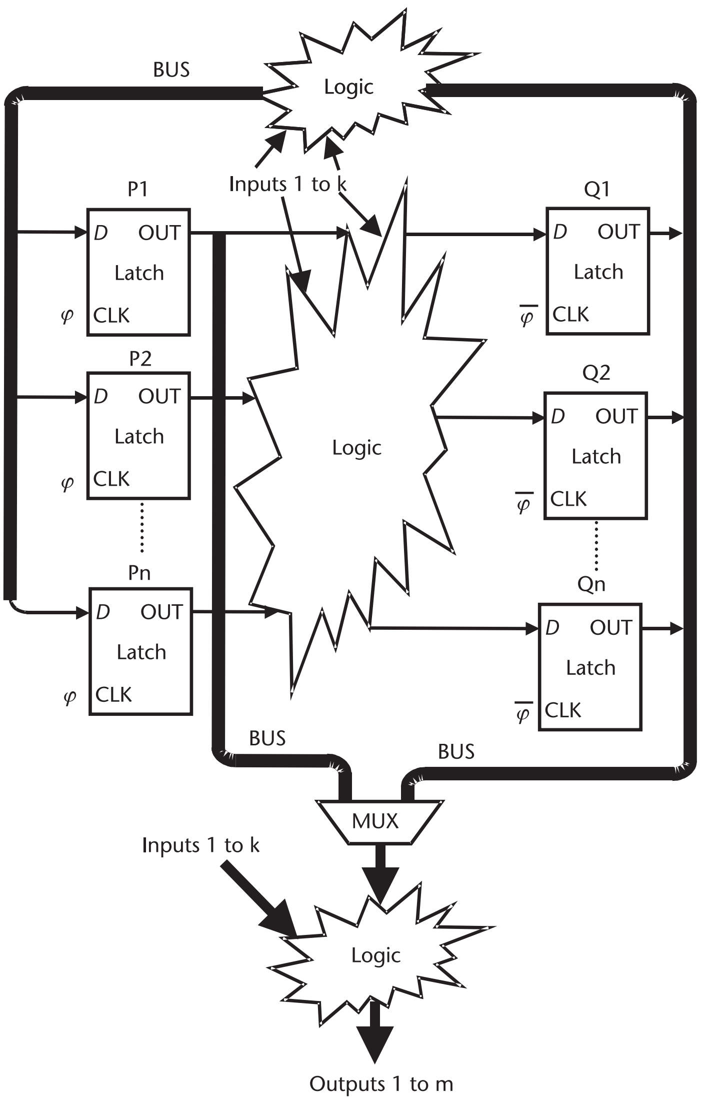


Figure 6.1 A generic latch logic circuit.

are not transparent (i.e., they are holding and are thus in one of the other two states: one or zero). Because the phase of the clock causes the latches to move between the transparent and holding states, the clock also contributes another bit to define the state of the system. Applying this to Figure 6.1, when the clock is high, the *P* latches will determine the state of the system; when the clock is low, the *Q* latches will determine the state of the system. Outputs of the system can be determined by the current state and the inputs to the system. In general, the output has to be connected through a multiplexer, also called a MUX in Figure 6.1. (A multiplexer is a switch that can be controlled to connect one of several inputs to the output, leaving the rest disconnected.) When the clock is high, the output is a function of only the system inputs and outputs from the *P* latches. When the clock is low, the output is a function of system inputs as well as the outputs from the

Q latches. When building these circuits, *Q* latches usually only feed back to *P* latches (through logic) but never to other *Q* latches, and vice versa. Throughout this chapter, latch logic is used to design examples of common divider circuits. Once these techniques are mastered, they can be used to make any divider required. In this chapter, these basic building blocks are then used as the basis for more complicated block-level dividers.

6.2.1 A Static Divide-by-Two Circuit

The first step in designing a divider circuit is to envision what output waveforms the circuit should produce. Once this is completed, the rest is mechanical. One of the most basic dividers is a static divide-by-two circuit. In such a divider, the output should be a repeated pattern alternating between being zero for two clock phases and one for two clock phases. An additional desirable feature would be to have two outputs separated by a phase shift of 90°. Such phase-shifted outputs can be very useful, for example, if the divider is needed to drive in-phase (I) and quadrature-phase (Q) mixers in a telecommunications application. Figure 6.2 shows a drawing of the desired outputs from this circuit. Note that this circuit has four states (two *P* states when the clock is high and two *Q* states when the clock is low). The states are assigned and labeled above the clock signals as $P = 0$, $Q = 0$, $P = 1$, and $Q = 1$. Note that uppercase labels *P* and *Q* are used for latches or for states, while lower case labels, *p*1, *q*1, are used for outputs from latches. In principle, state assignment is arbitrary. However, a good assignment will ensure that a simpler logic circuit is required to generate the outputs, which, in this case, must alternate between zero and one.

The state graph for such a divider, which can be made by studying Figure 6.2, is shown in Figure 6.3. Here, the circles represent states when the clock is high,

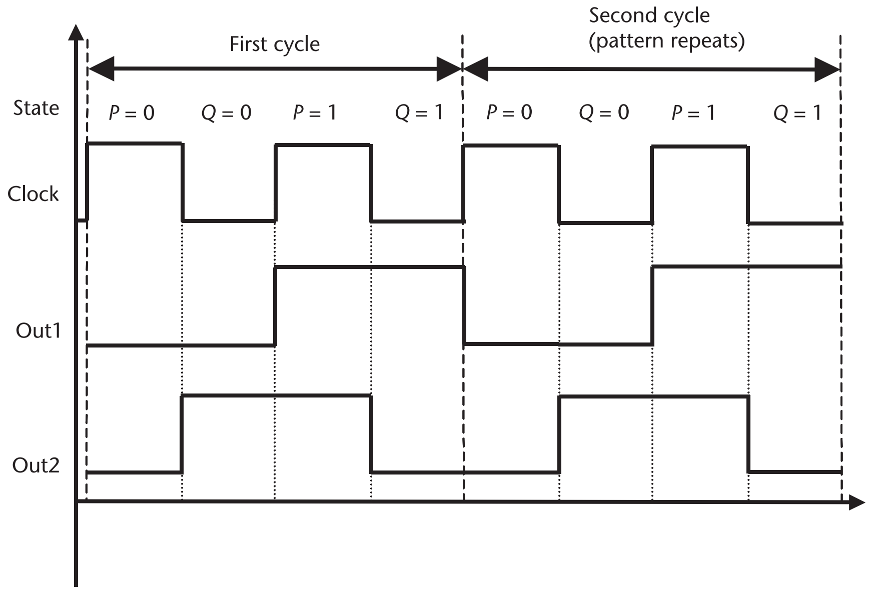


Figure 6.2 The outputs from a static divide-by-two circuit.

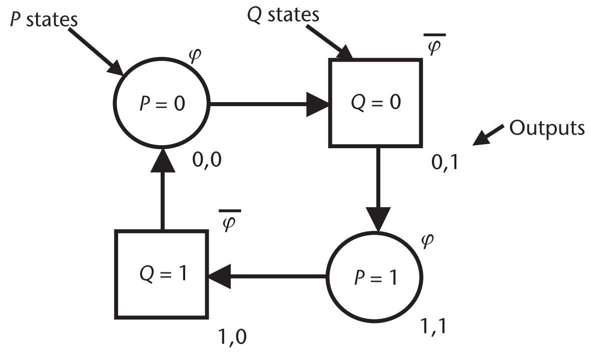


Figure 6.3 The state graph for a divide-by-two circuit. Circle states represent states when the clock is high (also marked with φ); square states represents states when the clock is low (marked with $\overline{\varphi}$).

while the boxes represent states when the clock is low. The outputs are shown next to their associated square or circle. Tracing out the state graph begins from a circle state, as shown in Figure 6.3. From there, the state changes to a square state, and the second output changes. Next, the circuit changes state back to a circle state, and the first output goes high. Then, a square state is the last state with the second output going low before it returns to the first state and the cycle repeats. Now each circle state must be assigned a unique binary number, or one at least unique from any other circle state but possibly the same as a square state. It is not necessary to have uniqueness between square and circle states, as the clock provides a way to tell them apart. The binary number for a state will correspond to the states of the *P* latches if the clock is high and the *Q* latches if the clock is low. Note that since this circuit has four states (two circle states and two square states), it will need one *P* latch and one *Q* latch, as illustrated in Figure 6.4. State assignment can be arbitrary, but making an effort to have as few state bits change as possible in as many cases as possible may simplify the logic. In the state assignment chosen here, there are two transitions where the binary number does not change at all from a circle state to the next square state.

From the completed state diagram, the state tables can be constructed. For this divider, there will be one state table for each clock phase, as shown in Tables 6.1

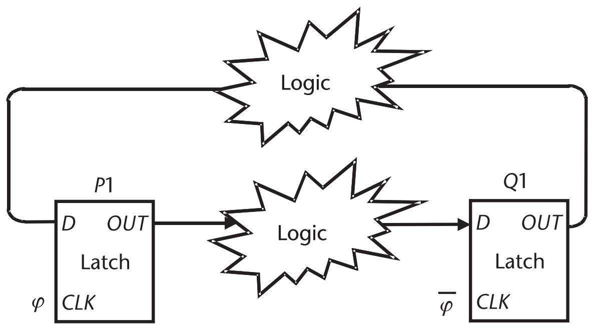


Figure 6.4 A logic circuit with four states (each latch gives a maximum of two states).

and 6.2. Note that we use the convention of the "+" sign to indicate that the variable value is for its next state.

Since there are no inputs to this circuit, the logic is rather trivial. If the clock is high, the next state is the inverse of the current state, and if the clock is low, then the next state is simply equal to the current state. The outputs can also be determined:

$$q1^{+} = p1 \tag{6.1}$$

$$p1^{+} = \overline{q1} \tag{6.2}$$

$$Out1(\varphi) = p1 \tag{6.3}$$

$$Out1(\overline{\varphi}) = q1 \tag{6.4}$$

$$Out2(\varphi) = p1 \tag{6.5}$$

$$Out2(\overline{\varphi}) = \overline{q1} \tag{6.6}$$

The resulting final circuit with the logic in place (only one inverter) is shown in Figure 6.5(a). This could also be thought of as a single flip-flop with its output inverted and fed back to its input. In this example, the input is the clock, and the output is a square wave at one-half the frequency of the input. The outputs on each phase of the clock are also shown in Figure 6.5(a).

It is desirable to get the outputs without any additional circuitry if possible. First, consider $Out1$. When the clock is high, $Out1$ is present at the output of $P1$. When the clock is low, $Out1$ is present on the output of $Q1$. However, also note that when the clock is high, $Q1$ is transparent, so the output of $P1$ also appears at the output of $Q1$. Thus, $Out1$ can be seen on either phase of the clock at the output of $Q1$, even when $Q1$ is transparent. Similarly, for $Out2$, when the clock is high, $Out2$ is present at the output of $P1$, but when the clock is low, $Out2$ is present at the output of the inverter. In this state, $Out2(\overline{\varphi})$ passes straight through $P1$ because $P1$ is transparent. Thus, $Out2$ can be seen at the output of $P1$ regardless of the phase of the clock. This simplification can be seen in Figure 6.5(b).

Table 6.1 State Table for a Divide-by-Two Circuit (Clock High)

State $p1$ (φ)	*Next State* $q1^{+}$ $(\overline{\varphi})$	*Outputs* (φ)
0	0	0,0
1	1	1,1

Table 6.2 State Table for a Divide-by-Two Circuit (Clock Low)

State $q1$ $(\overline{\varphi})$	*Next* $p1^{+}$ *State* (φ)	*Outputs* $(\overline{\varphi})$
0	1	0,1
1	0	1,0

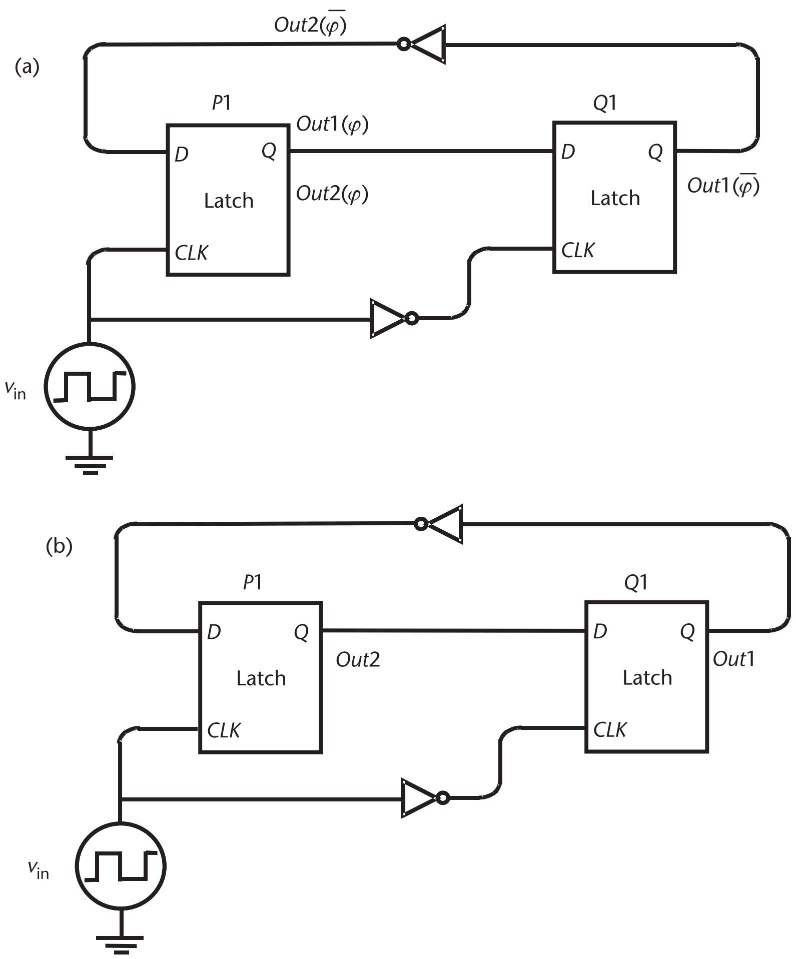


Figure 6.5 A divide-by-two circuit made from a *D*-flip-flop: (a) outputs on different clock phases, and (b) outputs without the need for a MUX.

6.2.2 Programmable Divide-by-Two or Divide-by-Three Circuit

Dividers that are more complicated can be designed using the methods outlined in the previous section. For instance, suppose a circuit is needed that sometimes divides by two and at other times divides by three, depending on the value of an input (Div). The output of such a circuit might look like that shown in Figure 6.6. This circuit does not have a 50% duty cycle when it is in divide-by-three mode ($Div = 1$); rather, it is high for two clock phases and low for four clock phases before the cycle repeats. From Figure 6.6, the output state diagram shown in Figure 6.7 can be derived as discussed previously. Note that the path through the state diagram depends on the value of the input. If the input is low, then the state table follows the pattern of the divide-by-two circuit just discussed, but if the input is high, then an extra half-cycle of delay is added to the circuit, forcing it to divide by three rather than two. Also note that since the output is not a 50% duty cycle square wave when dividing by three, the need for extra logic attached at the output is eliminated, as will be shown later in this section. It is important that the state

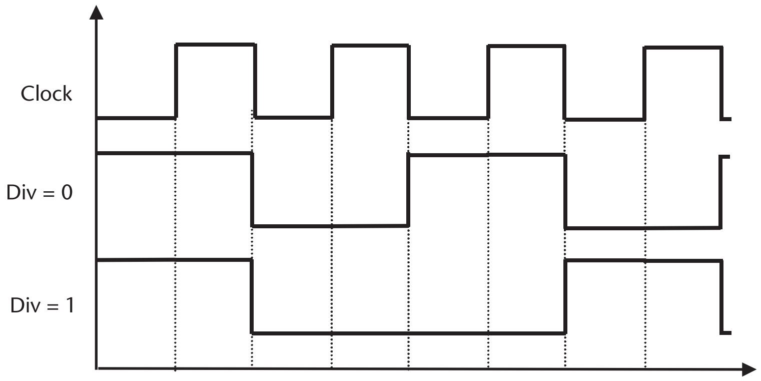


Figure 6.6 Desired outputs of the divide-by-2/3 circuit.

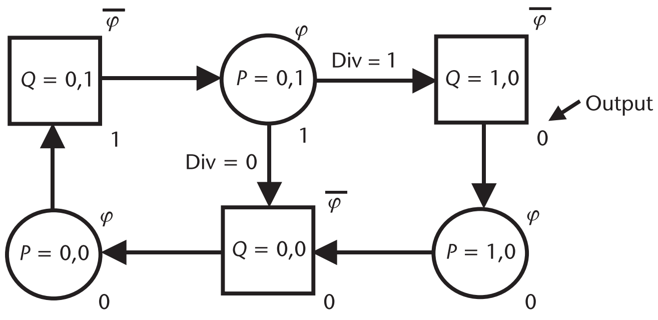


Figure 6.7 A divide-by-2/3 state graph.

assignments be done very carefully in this circuit to minimize the number of state variable changes for each transition between a $\overline{\varphi}$ state and an adjacent φ state. This will lead to a minimum amount of logic between the latches. In this circuit, a maximum of one state variable was changed in each transition, except when moving from the φ state 0,1 to the $\overline{\varphi}$ state 1,0 when Div = 1 in Figure 6.7. For this transition, two state-variable changes were unavoidable.

In Figure 6.7, 2 bits are required to define three states completely, as there are three circle and three square states. Therefore, two Q latches are needed to define three unique Q states, and two P latches will be required to define three unique P states. The four latches are labeled $Q1$, $Q2$, $P1$, and $P2$, as shown in Figure 6.8. The outputs from these latches will be called $p1$, $p2$, $q1$, and $q2$.

State tables for this circuit can now be constructed and are shown in Tables 6.3 and 6.4.

From these two tables, six Karnaugh maps must be constructed and will be shown in Tables 6.5 through 6.11. Note that Karnaugh maps show outputs for all combinations of inputs, with inputs ordered in such a way that only one bit ever changes in adjacent positions. The Karnaugh map allows for easy determination of the simplest logic to realize the function. Table 6.5 shows the Karnaugh map for $p1^+$.

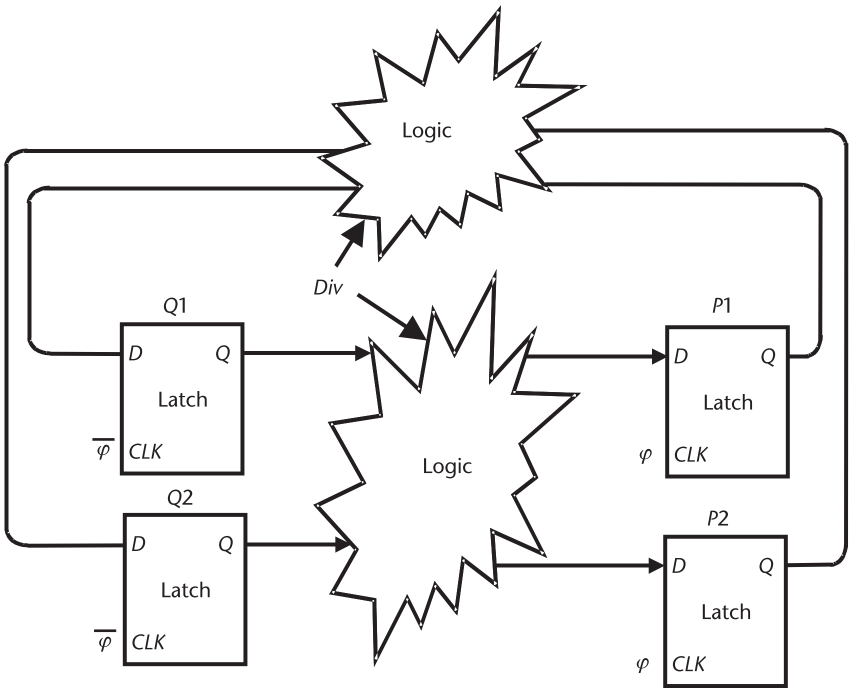


Figure 6.8 A divide-by-2/3 circuit under construction.

Table 6.3 State Table for a Divide-by-2/3 Circuit (Clock Low)

State ($\overline{\varphi}$)	*Next State* (φ)		*Output* ($\overline{\varphi}$)	
	$p1^+p2^+$	$p1^+p2^+$		
$q1q2$	*Div = 0*	*Div = 1*	*Div = 0*	*Div = 1*
00	00	00	0	0
01	01	01	1	1
11	d/c	d/c	d/c	d/c
10	10	10	d/c	0

Table 6.4 State Table for a Divide-by-2/3 Circuit (Clock High)

State (φ)	*Next State* ($\overline{\varphi}$)		*Output* (φ)	
	$q1^+q2^+$	$q1^+q2^+$		
$p1p2$	*Div = 0*	*Div = 1*	*Div = 0*	*Div = 1*
00	01	01	0	0
01	00	10	1	1
11	d/c	d/c	d/c	d/c
10	00	00	d/c	0

Note that d/c = don't care, which means that the next state or output is set arbitrarily.

Therefore, the equation for $p1^+$ is given by

$$p1^+ = q1 \tag{6.7}$$

The Karnaugh maps for $p2^+$, Out, $q1^+$, and $q2^+$ can similarly be constructed.

Table 6.5 Karnaugh Map for $p1^+$

$q1q2$	*Div = 0*	*Div = 1*
00	0	0
01	0	0
11	d/c	d/c
10	1	1

Table 6.6 Karnaugh Map for $p2^+$

$q1q2$	*Div = 0*	*Div = 1*
00	0	0
01	1	1
11	d/c	d/c
10	0	0

Table 6.7 Karnaugh Map for *Out* (Clock Low)

$q1q2$	*Div = 0*	*Div = 1*
00	0	0
01	1	1
11	d/c	d/c
10	d/c	0

Table 6.8 Karnaugh Map for $q1^+$

$p1p2$	*Div = 0*	*Div = 1*
00	0	0
01	0	1
11	d/c	d/c
10	0	0

Table 6.9 Karnaugh Map for $q2^+$

$p1p2$	*Div = 0*	*Div = 1*
00	1	1
01	0	0
11	d/c	d/c
10	0	0

Table 6.10 Karnaugh Map for *Out* (Clock High)

$p1p2$	*Div = 0*	*Div = 1*
00	0	0
01	1	1
11	d/c	d/c
10	d/c	0

Table 6.11 Karnaugh Map for *Out* ($\overline{\varphi}$)

$q1q2$	*Div = 0*	*Div = 1*
00	0	0
01	1	1
11	d/c	d/c
10	d/c	1

From the Karnaugh maps, the following additional equations can be derived:

$$p2^+ = q2 \tag{6.8}$$

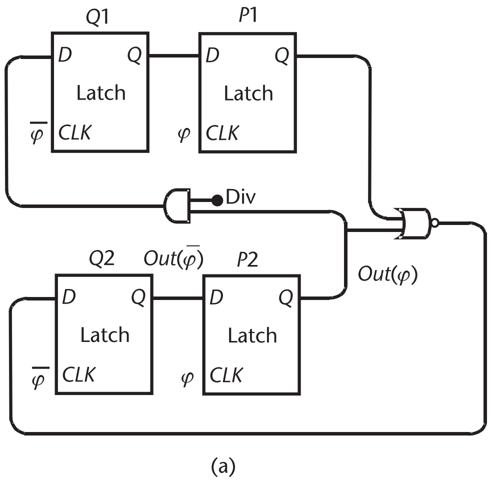
$$Out(\overline{\varphi}) = q2 \tag{6.9}$$

$$q1^+ = Div \cdot p2 \tag{6.10}$$

$$q2^+ = \overline{p1} \cdot \overline{p2} = \overline{p1 + p2} \tag{6.11}$$

$$Out(\varphi) = p2 \tag{6.12}$$

From (6.9) and (6.12), the equivalent circuit logic can be added to Figure 6.8 and is shown in Figure 6.9(a). It can be seen that the output has a value of $q2$ when the clock is low and a value of $p2$ when the clock is high. When the clock



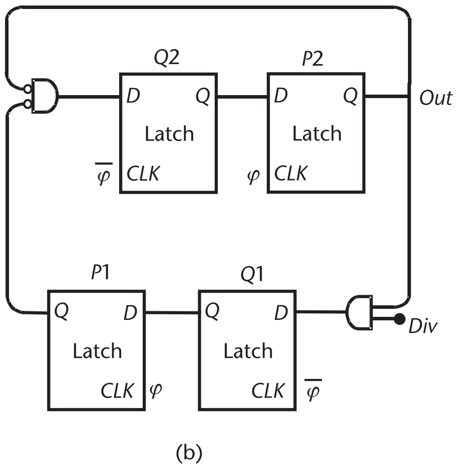


Figure 6.9 A completed divide-by-2/3 circuit drawn in (a) and redrawn in (b).

is low, $P2$ is transparent; therefore, the output present at the output of $Q2$ also appears at the output of $P2$ during this clock phase. Thus, the output can always be taken at the output of $P2$ on either clock phase. An equivalent version of the circuit is shown in Figure 6.9(b) without the need to specify the clock phase for the output.

It is also possible, with a trivial modification, to make this circuit have a 50% duty cycle when in divide-by-three mode. This requires one modification to the state diagram in Figure 6.7. Since the output is high for two clock phases and low for four clock phases, one of the low outputs must be converted to a high output. Selecting the $\overline{\varphi}$ state 1,0 as the output that should remain high changes Table 6.7 into Table 6.11.

In this case, the output equations are given by

$$Out(\overline{\varphi}) = q1 + q2 \tag{6.13}$$

$$Out(\varphi) = p2 \tag{6.14}$$

Thus, in this modified divider, the output can no longer be taken directly from the output of the $P2$ latch. The circuit must therefore have an OR gate added to it for the output to have a 50% duty cycle in all cases. Note that in the case of the $\overline{\varphi}$ states, an OR gate connected between the outputs of $Q1$ and $Q2$ will give the correct output, but one connected to $Q1$ and $P2$ will also work as $P2$ is transparent in the $\overline{\varphi}$ state and, therefore, has the same value as $Q2$. Now, in the φ states, $P2$ is the desired output, but the value of $P2$ is also present at the output of $Q1$ when Div is high, and the output of $Q1$ is low if Div is low in these states. Therefore, having an OR gate connected to $Q1$ and $P2$ will not affect the output value for the φ states. Thus, the addition of a single OR gate can be used to generate a 50% duty cycle square wave output for all cases. The resulting circuit is shown in Figure 6.10.

6.2.3 A 50% Duty Cycle, High-Speed, Divide-by-Three Circuit

An alternative divide-by-three circuit can be constructed using the latch with the invertible clock discussed in Chapter 5. This circuit uses only three latches rather than four [1]. The three cascaded, modified latch circuits are set up to generate a Johnson counter as shown in Figure 6.11.

In this circuit, on each phase of the clock, one latch holds, and the other two are transparent. Thus, zeros and ones rotate through, as will now be illustrated. Suppose that, initially, the output of each latch is zero. Since latch A has one as an input (from the inverter) and an output of zero, latch A cannot be transparent. However, the other two latches have both inputs and outputs of zero and, therefore, can be, and are, transparent. Thus, the inputs θ need to be set so that, regardless of the phase of the clock, the appropriate latches are always either holding or transparent, as needed. With this information, a state table can be constructed (see Table 6.12).

From here, we can construct a Karnaugh map for each θ. These are shown in Tables 6.13 through 6.15.

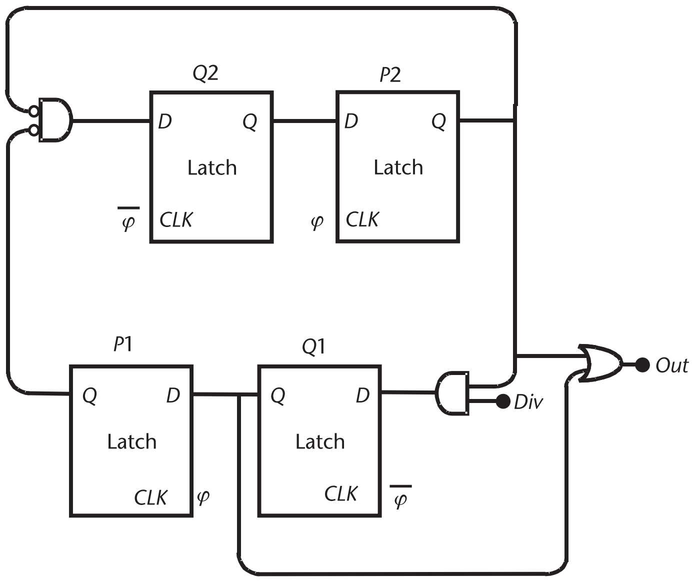


Figure 6.10 A completed divide-by-2/3 circuit with 50% duty-cycle output.

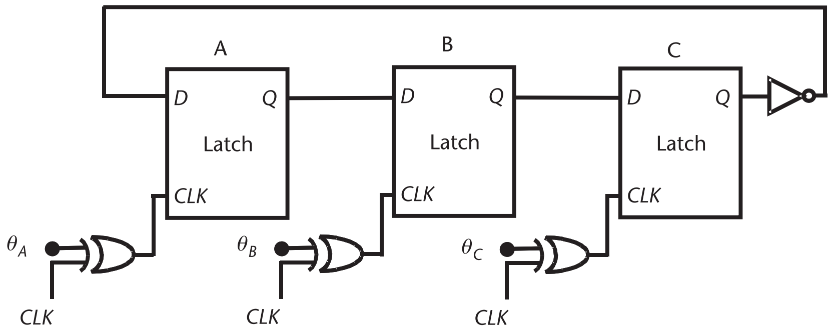


Figure 6.11 A Johnson counter divide-by-three circuit using modified latches.

Table 6.12 State Table for a Divide-by-Three Circuit

		Latch Value			*Holding Input*		
State	*Clock*	*A*	*B*	*C*	θ_A	θ_B	θ_C
1	0	0	0	0	0(L)	1(T)	1(T)
2	1	1	0	0	0(T)	1(L)	0(T)
3	0	1	1	0	1(T)	1(T)	0(L)
4	1	1	1	1	1(L)	0(T)	0(T)
5	0	0	1	1	1(T)	0(L)	1(T)
6	1	0	0	1	0(T)	0(T)	1(L)

Note: L = latched; T = transparent.

Table 6.13 Karnaugh Map for θ_A

A B	*C = 0*	*C = 1*
00	0	0
01	d/c	1
11	1	1
10	0	d/c

Table 6.14 Karnaugh Map for θ_B

A B	*C = 0*	*C = 1*
00	1	0
01	d/c	0
11	1	0
10	1	d/c

Table 6.15 Karnaugh Map for θ_C

A B	*C = 0*	*C = 1*
00	1	1
01	d/c	1
11	0	0
10	0	d/c

Thus, the θ inputs are given by

$$\theta_A = \mathrm{B} \tag{6.15}$$

$$\theta_B = \overline{\mathrm{C}} \tag{6.16}$$

$$\theta_C = \overline{\mathrm{A}} \tag{6.17}$$

Therefore, the θ inputs can be wired using the outputs from the latches and almost no additional logic. The final divider is shown in Figure 6.12.

6.2.4 A Multimodulus Divider

The state graph for a modified divide-by-2/3 cell that can be used to make an MMD is shown in Figure 6.13. The idea is that these cells will be cascaded, which means that additional signals will be required to interface between consecutive stages [2, 3]. Note that this divider is very similar to one discussed previously, except that there are now two inputs, Mod_{in} and R, that have to be high for divide-by-three. In addition, there is one more output called Mod_{out}.

The new divide-by-2/3 circuit can be designed using the methods already outlined. Skipping a few steps, the final circuit is shown in Figure 6.14. This circuit is the same as that shown in Figure 6.9 except for the additional AND gate for loading the division ratio R.

The divide-by-2/3 cell divides the input frequency by two or three, depending on both Mod_{in} and R. When Mod_{in} is low, regardless of the value of R, the output

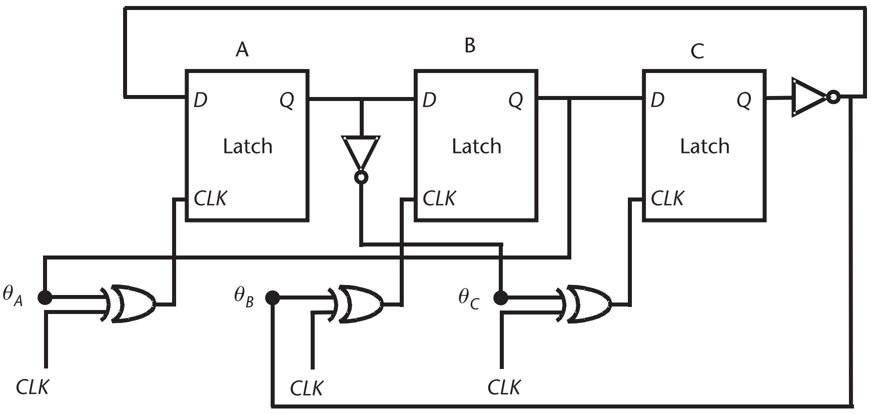


Figure 6.12 A Johnson counter divide-by-three circuit using modified latches completed with logic added.

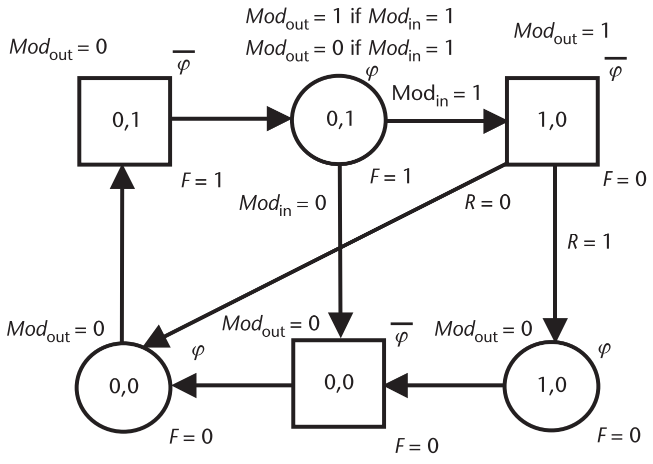


Figure 6.13 State graph for a modified divide-by-2/3 with two controlling inputs, Mod_{in} and R, both of which must be high for divide-by-three. Outputs are Mod_{out} and F.

of $P1$ will be low, which means that the bottom half of the circuit in Figure 6.14 does not affect the state of the system, and the remaining top half of the circuit is simply a divide-by-two circuit that was derived earlier. Figure 6.15 shows the waveforms generated by the circuit in Figure 6.14 when Mod_{in} is high. The first half of the figure shows the case when R is low. In this case, the output of $P1$ is always low, and the divider will be in divide-by-two mode. When both Mod_{in} and R are high, the circuit divides by three, just as the circuit described in the last section. In this case, the two latches on the bottom row cause the circuit to swallow an additional input pulse. The Mod_{out} signal has the same frequency as that of the output signal F, yet is high for only one input cycle.

The divider just discussed can be cascaded to make higher-order dividers. For instance, consider two of the previous circuits connected as shown in Figures 6.16 and 6.17, where Mod_{out} of the second divider is used to drive Mod_{in} of the first divider. The last cell in such a chain of dividers always has its Mod_{in} input connected

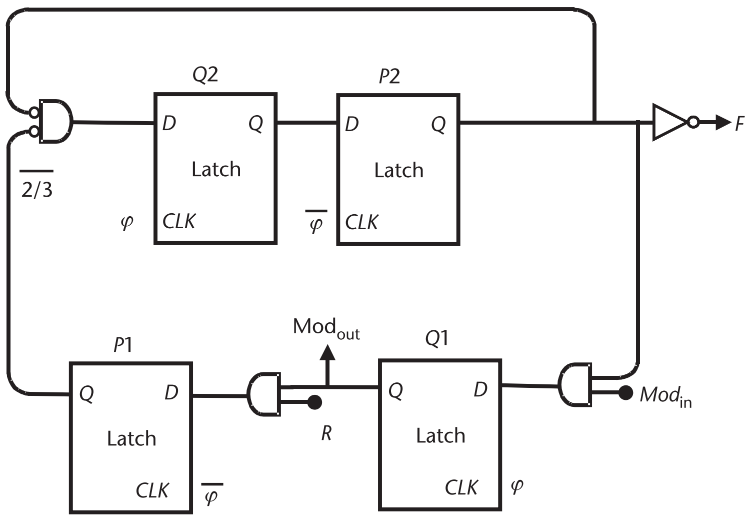


Figure 6.14 A divide-by-2/3 circuit with a *Mod*$_{in}$ control and a *Mod*$_{out}$ signal added.

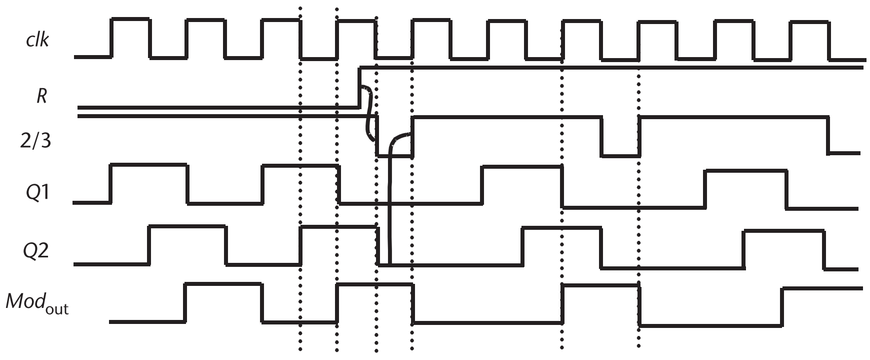


Figure 6.15 Waveforms of the divide-by-2/3 circuit, assuming that *Mod*$_{in}$ is high.

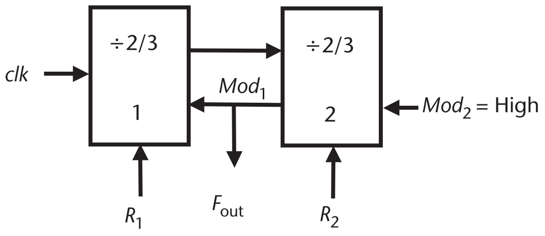


Figure 6.16 A divide-by-four-to-seven made from two 2/3 stages.

high. If R_1 is low, then the first stage always divides by two; therefore, the second stage gets a 50% duty cycle square wave at half the input frequency, regardless of the value of Mod_1. Since Mod_2 is always high, the second divider then divides this signal either by two, if R_2 is low, or by three, if R_2 is high. Thus, if R_1 is low, the system divides by four when R_2 is low and by six when R_2 is high.

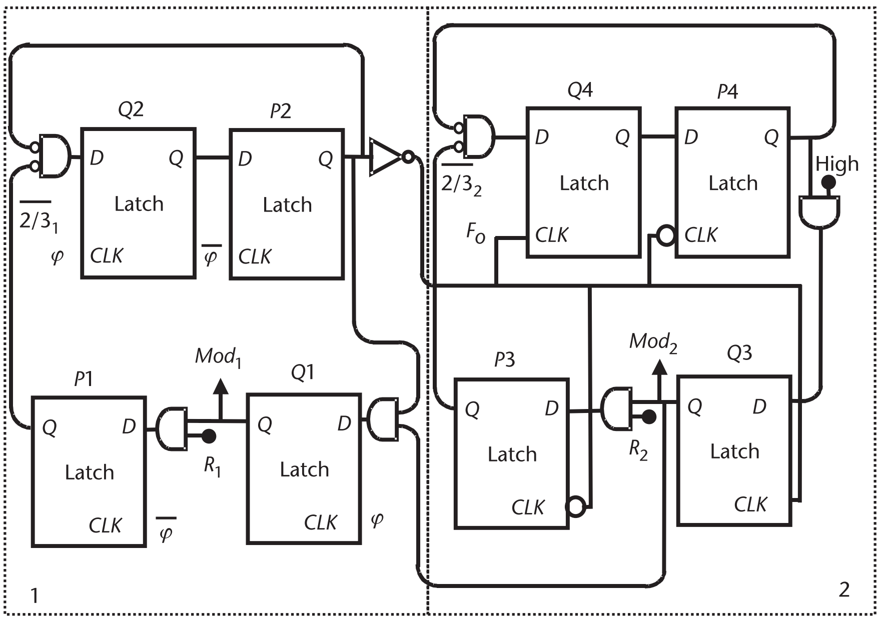


Figure 6.17 A divide-by-four-to-seven made from two 2/3 stages shown at latch level.

When R_1 is high, the circuit behavior is more complicated because Mod_1 will come into play. The case of $R_1 = 1$ and $R_2 = 0$ is shown in Figure 6.18. In this case, the second divider will always divide by two, but the Mod_2 signal will be high every second cycle of F_o. Thus, for one complete cycle of the output, the first divider will swallow one extra pulse (divide by three once), causing the whole circuit to divide by five. Likewise, it can be shown that with both inputs set to one, divide-by-seven is achieved as illustrated in Figure 6.19. In this case, the second

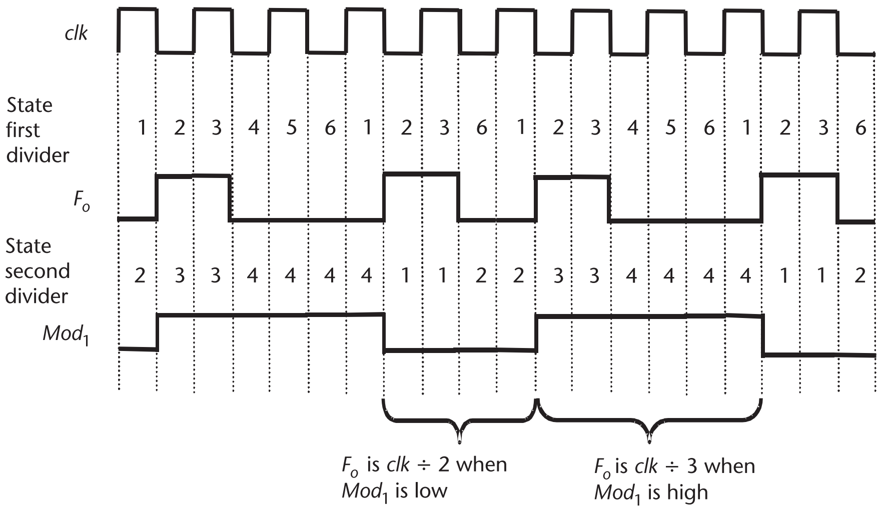


Figure 6.18 Waveforms for two cascaded divide-by-2/3 cells in divide-by-five mode (R_1 = high; R_2 = low).

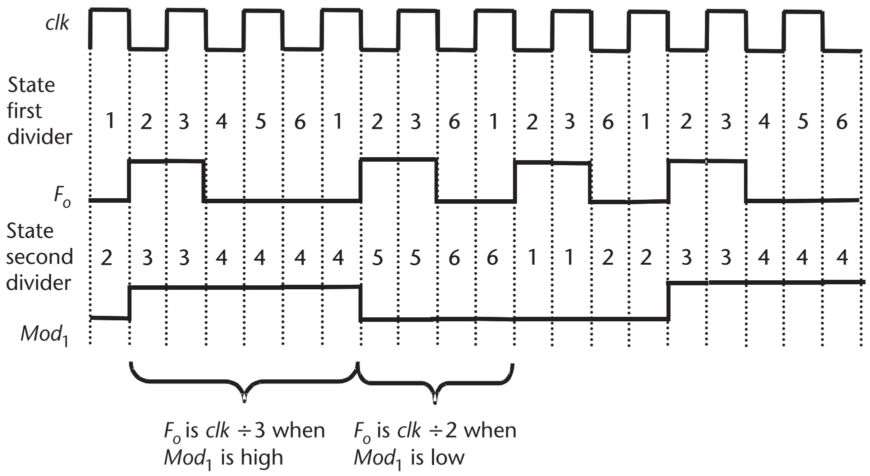


Figure 6.19 Waveforms for two cascaded divide-by-2/3 cells in divide-by-seven mode (R_1 = high, and R_2 = high).

divider always divides by three, and, once again, the Mod_1 signal causes the first divider to swallow one extra pulse every output cycle of the second divider. This adds one input period to the period of the output signal. Thus, in this circuit, to produce one output clock cycle, a minimum of four input clock cycles is required. If R_2 is high, the second block causes the circuit to swallow two additional input pulses, and if R_1 is high, the first block will swallow one additional input pulse. Thus, the total output period is given by

$$T_{\text{out}} = T_{\text{in}}R_1 + 2T_{\text{in}}R_2 + 4T_{\text{in}} \tag{6.18}$$

In general, as many divide-by-2/3 cells as necessary can be cascaded to make higher-order dividers, as shown in Figure 6.20. Note that when implementing each cell 1 through n, the nth cell runs at a speed of $1/n$ or less of the first cell. Therefore, if each cell is implemented as a custom block, the current can be scaled accordingly to achieve low-power operation. When connected in a ripple chain as shown in Figure 6.20, the last cell always has its Mod_{in} input as logic high, and for all other cells, the Mod_{in} input is high for only one of its output cycles during one output cycle of the following stage. In general, the output period is given by

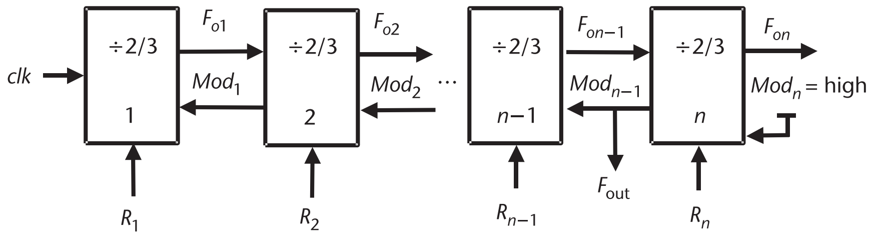


Figure 6.20 An MMD made of programmable divide-by-2/3 stages.

$$T_{\text{out}} = T_{\text{in}}R_1 + T_{\text{in}}2^1R_2 + \ldots + T_{\text{in}}2^{n-2}R_{n-1} + T_{\text{in}}2^{n-1}R_n + T_{\text{in}}2^n \tag{6.19}$$

Therefore, the division ratio is given by

$$N = R_1 + 2^1R_2 + \ldots + 2^{n-2}R_{n-1} + 2^{n-1}R_n + 2^n \tag{6.20}$$

Thus, for this type of divider, a division ratio that increases in unity integer steps from 2^n to $2^{n+1} - 1$ is achieved. With additional OR gates, wider-range division ratios can be obtained, as illustrated in [2].

6.2.5 A Generic MMD Architecture

The MMD architecture with all 2/3 cells may not necessarily end up with minimum gate count and power consumption. This section discusses a generic MMD design algorithm, along with the implementation of each cell, which takes the minimum hardware and current [4]. The MMD architecture discussed in the previous section, based on [2], is a special case of the generic algorithm. When the MMD division range is not large compared to its minimum division ratio, the generic algorithm will result in an optimal MMD architecture that is different from the one given in [2]. For the case of a large MMD division range, the generic algorithm leads to the architecture with all divide-by-2/3 cells, as discussed above.

The generic MMD architecture includes a number of divide-by-2/3, dual-modulus cells cascaded with a divide-by-($P/P + 1$) dual-modulus cell (P being an integer and $P \geq 2$) in a ripple fashion as shown in Figure 6.21. Here only the last cell is selected to be a divide-by-($P/P + 1$) cell so that all of the division ratios in the required range can be programmed with a unit step increment. If any one of the preceding cells is not a 2/3 cell, then the unit step increment is not guaranteed. If a step increment other than one is desired, the optimal architecture is to place a fixed-ratio, divide-by-S stage in front of the MMD cells, as shown, so that the MMD has a programmable step size of S. The architecture shown in Figure 6.21 provides the division ratio as

$$N = \left(2^{n-1}P + 2^{n-1}R_{n-1} + 2^{n-2}R_{n-2} + \ldots + 2^1R_1 + R_0\right) \cdot S \tag{6.21}$$

where $R_0, R_1, \ldots R_{n-2}, R_{n-1}$ are the programmable MMD control bits.

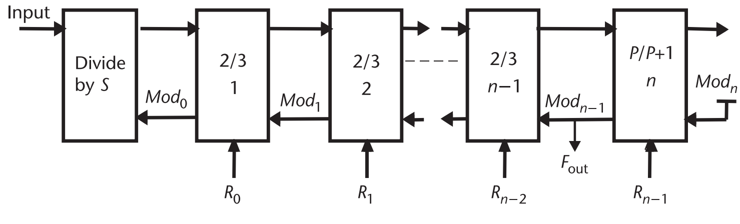


Figure 6.21 A generic MMD architecture.

It can be observed from (6.21) that the last cell, being $P/P+1$, increases the minimum division ratio while maintaining the unit step increment for the MMD. To determine the number of cells needed and to select a proper division ratio of P, the following generic algorithm should be followed:

1. Assume that the required division ratios are from $D_{\min}$ to $D_{\max}$; then, the number of divisor steps is given by

$$\text{Number of divisor steps} = D_{\max} - D_{\min} + 1 \tag{6.22}$$

2. If the required range is greater than the minimum division ratio, $D_{\min}$, the MMD should be constructed using an architecture with all divide-by-2/3 cells.
3. The implemented MMD range, defined from M to N, may be larger than the required range $D_{\min}$ to $D_{\max}$. Initially, however, the minimum implemented division ratio M is set to $D_{\min}$.
4. Now the number of cells required becomes

$$n = \lceil \log_2(D_{\max} - M + 1) \rceil \tag{6.23}$$

where the function $\lceil a \rceil$ denotes rounding the number a to the nearest integer larger than a.

5. The division ratio for the last cell can be found from

$$P = \lfloor M/2^{n-1} \rfloor \tag{6.24}$$

where the function $\lfloor a \rfloor$ denotes rounding the number a to the nearest integer less than a.

6. If $M/2^{n-1}$ is not an integer, then reset $M = P \cdot 2^{n-1}$, and go to step 4.
7. If $M/2^{n-1}$ is an integer, it is necessary to evaluate whether using a single $P/P+1$ cell or using any combination of cascaded cells, such as $2/3 \rightarrow \lfloor P/2 \rfloor/(\lfloor P/2 \rfloor + 1)$, or $2/3 \rightarrow 2/3 \rightarrow \lfloor P/4 \rfloor/(\lfloor P/4 \rfloor + 1)$, . . . , or using all 2/3 cells will achieve lower current consumption and smaller die size.
8. The final MMD architecture is thus a combination of stages, as shown in Figure 6.21. If only 2/3 cells are used, then the total number of cells required is

$$n_{2/3} = \lceil \log_2(D_{\max} + 1) \rceil - 1 \tag{6.25}$$

Example 6.1: Use of the Algorithm to Determine the Optimum Divider Design

The division ratios from $D_{\min} = 102$ to $D_{\max} = 108$ are required. Determine how to implement an MMD to perform this function.

Solution: Step 1 indicates that there will be a total of seven divisor ratios. Step 2 indicates that the divider should not be designed with all 2/3 cells. In step 3, the implemented MMD design range initially starts from $M = 102$, and the number of cells required is $n = 3$ as outlined in step 4. Then, the division ratio of the last cell becomes $P = 25$ (step 5), and M is reset to 100 (step 6). Next, the number of cells required is recalculated as $n = 4$ (returning to step 4), the division ratio of the last cell is 12 (step 5), and M is set to 96 (step 6). On the next iteration of steps 4 to 6, the values of n, P, and M remain the same, so we continue to step 7. Since the implementation of a 12/13 cell requires less hardware and consumes less current than the combination of a 2/3 cell cascaded with a 6/7 cell, the last cell-division ratio is maintained at $P = 12$. Thus, the final MMD architecture for this example is given by $(2/3) \rightarrow (2/3) \rightarrow (2/3) \rightarrow (12/13)$.

For a required division ratio from 68 to 108, the algorithm results in an MMD architecture with six 2/3 cells cascaded (i.e. $P = 2$), which is the case presented in [2].

The divide-by-2/3 cell can be implemented as shown in Figure 6.14, and a similar implementation can be extended to any $P/P + 1$ cell. However, the logical implementation of the cell differs for even- and odd-numbered P. For instance, the logical implementation for divide-by-4/5, -5/6, -6/7, and -8/9 cells are shown in Figures 6.22 through 6.25. These dual-modulus cells are formed by two rows of circuits, a basic divide-by-P circuit shown in the top row and two latches at the bottom for additional pulse swallowing to implement the divide-by-$(P + 1)$ function and for modulus control output. Since a divide-by-P circuit is always needed, it is more economical for P to be an even number. For instance, divide-by-5/6 and divide-by-6/7 cells require the same number of gates, yet divide-by-6/7 achieves a higher division ratio. The gate that has Mod_{in} as its input can be avoided because the Mod_{in} of the last cell is always logic high. Therefore, the output of the last latch on the top row of latches can be directly connected to the D input of the right latch on the bottom row. For 2/3 cells, the Mod_{out} signal has the same frequency as that of the F_o signal and is high for only one input cycle, as shown in Figure 6.15. For 4/5 cells, the output of the bottom right latch has twice the F_o

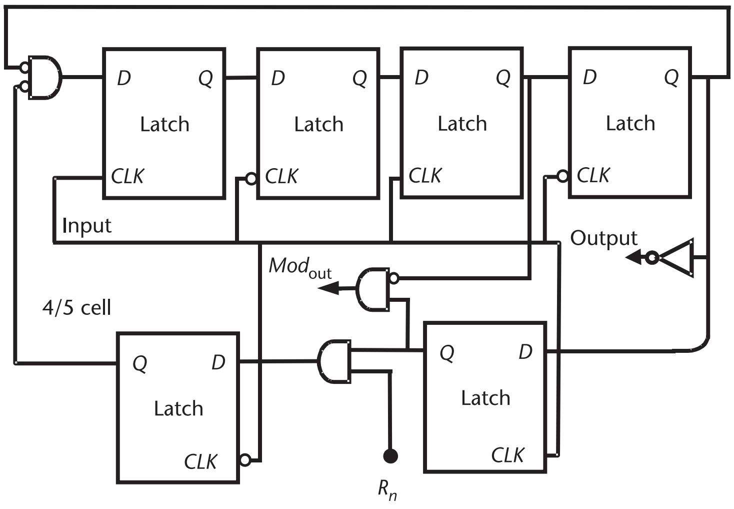


Figure 6.22 Latch-level schematic of a divide-by-4/5 cell for a cascaded MMD application.

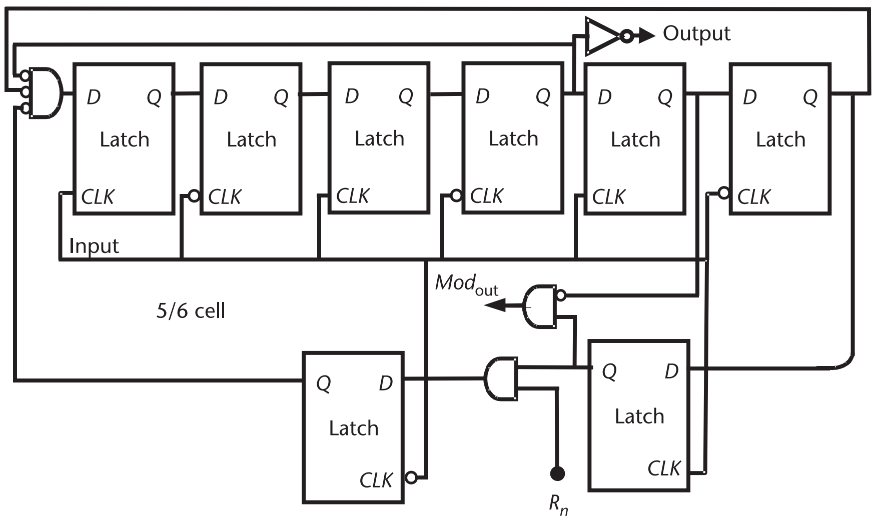


Figure 6.23 Latch-level schematic of a divide-by-5/6 cell for a cascaded MMD application.

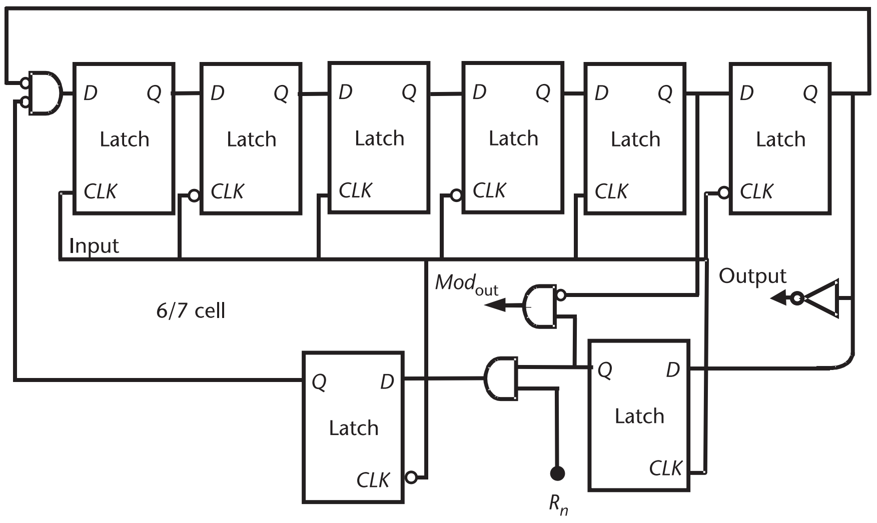


Figure 6.24 Latch-level schematic of a divide-by-6/7 cell for a cascaded MMD application.

frequency, since F_o is clocked by the input signal that has four times the F_o frequency. To obtain a Mod_{out} signal that is at the same frequency as F_o and remains high only for one input cycle, another AND gate is used in Figure 6.22 to remove one pulse for the Mod_{out} signal.

Example 6.2: Design of a Divider for a WLAN Synthesizer
Apply the generic MMD algorithm to design a divider for a direct downconversion radio for the WLAN 802.11a band.

Solution: We note that WLAN IEEE 802.11a occupies part of the Unlicensed National Information Infrastructure band, with the first channel center frequency

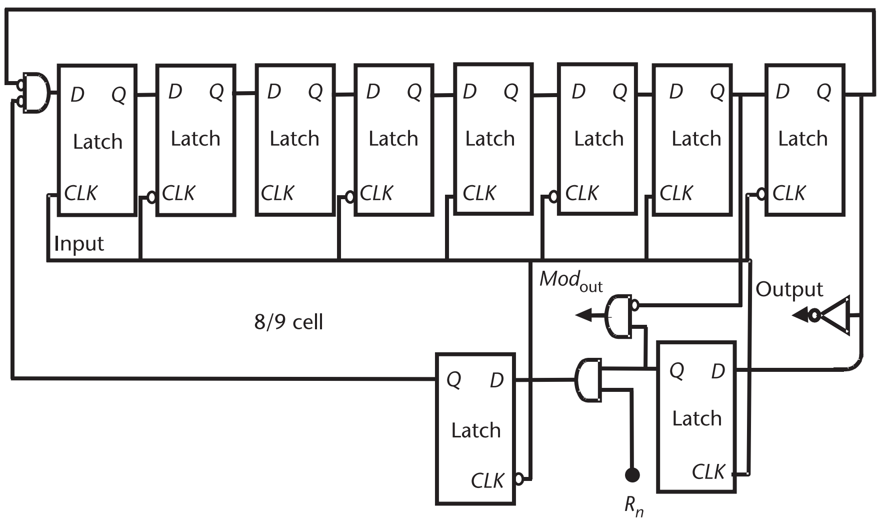


Figure 6.25 Latch-level schematic of a divide-by-8/9 cell for a cascaded MMD application.

at 5.18 GHz and the last channel center frequency at 5.32 GHz, and each channel occupies 20 MHz of bandwidth. Thus, the channel frequency (which is also the VCO frequency in a direct downconversion radio) is $F_{CH} = F_{VCO} = 5{,}180 + 20k$ ($k = 0, 1, 2 \ldots 7$) in megahertz. If a 40-MHz reference source is used, the MMD division ratio needs to be

$$N = \frac{F_{VCO}}{40} = 129 + \frac{(k+1)}{2} = 129.5 \text{ to } 133$$

Assuming that a second-order ΣΔ noise shaper with 2-bit output is used to remove the fractional spurs (the details of ΣΔ modulation will be given in Chapter 9), the instantaneous MMD division ratio could vary from –1 to 2 around its average division ratios. Thus, we need to have divider ratios at least one less than the minimum number and two greater than the maximum number, resulting in an MMD division range for IEEE 802.11a bands of N = 128 to 135. Applying the generic MMD algorithm discussed previously, the MMD can be realized using the cells 2/3 → 2/3 → 2/3 → 2/3 → 8/9, which requires 36 latches and 15 gates. Had an MMD architecture with all divide-by-2/3 cells been used, 42 latches and 21 gates would have been required, which leads to larger die size and more power consumption. The example for the IEEE 802.11a application points out that the division range of an MMD for a fractional-N synthesizer is often small with respect to its minimum division ratio. Hence, the generic MMD design algorithm provides a better architecture with less logic and lower power consumption than an MMD design using straight divide-by-2/3 cells. Output waveforms are shown for the division ratio of 135 of the IEEE 802.11a band in Figure 6.26. The Mod_{out} waveforms of all the cells have the same frequency with different duty cycles. Note that the Mod_4 signal coming from the 8/9 cell has the same frequency as the output frequency of the 8/9 cell due to the additional AND gate shown in Figure 6.25. Other Mod signals coming from 2/3 cells in the chain already have the same

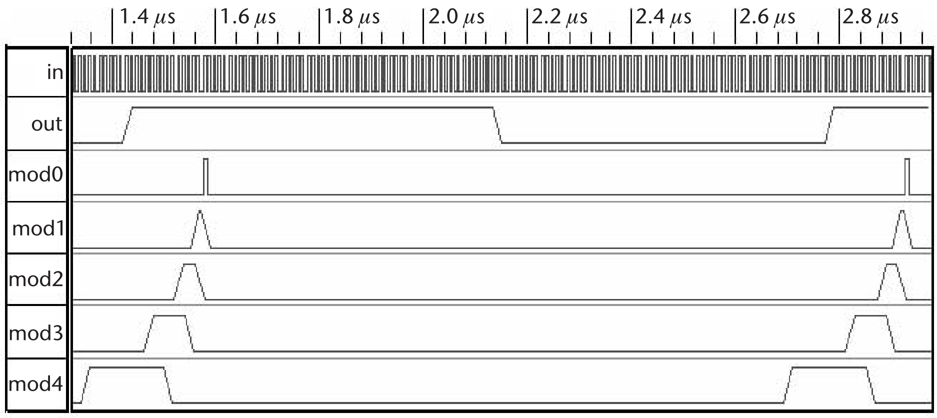


Figure 6.26 Waveforms of a 2/3 → 2/3 → 2/3 → 2/3 → 8/9 MMD architecture with division ratio 135.

frequency as the final output frequency. Hence, no additional AND gate is needed in the 2/3 cells for Mod_{out} signal generation. The period of the input signal is 10 ns, and, correspondingly, the outputs have periods of 1,350 ns. For clarity, only one cycle of the outputs has been shown.

6.2.6 Pulse-Swallow Dividers

The MMD outlined in the previous section is by no means the only one. There are many other modulus dividers, some of which are more common. Another widely used modulus divider architecture is called a pulse-swallow divider, an example of which is shown in Figure 6.27. The pulse-swallow divider includes a programmable frequency divider M, a dual-modulus prescaler $P/P + 1$, and a down counter A. The dual-modulus prescaler divides by either P or $P + 1$, depending on the value of the input *Modulus Control*. The *Programmable Counter* is a frequency divider with programmable division ratio M. The programmable divider differs from the modulus divider in the sense that, once it is programmed, the division ratio remains constant. In contrast, the division ratio of a modulus divider varies

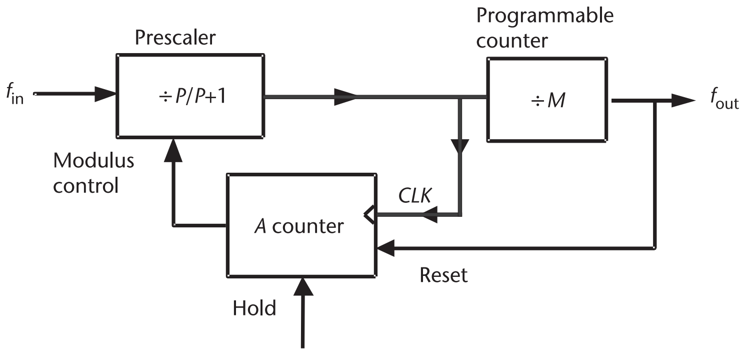


Figure 6.27 A pulse-swallow-modulus divider made of a divide-by-($P/P+1$) dual-modulus prescaler.

dynamically based on the *Modulus Control* signal. Since this divider is usually toggled at the VCO frequency, it requires much faster circuits than the other dividers shown in Figure 6.27. This divider is often called a prescaler because it directly follows the VCO, slowing down the VCO output frequency; thus, the programmable dividers do not need to operate at high speed. The M divider and A counter can normally be built using standard digital CMOS logic techniques, while the VCO and the dual-modulus prescaler are normally custom designed, taking into account analog design issues. A dual-modulus prescaler can achieve the same speed as an asynchronous divider by limiting the high-speed section to only one divide-by-two flip-flop, as discussed in [5].

The pulse-swallow divider operates in the following manner:

1. The M divider divides the output frequency of the dual-modulus prescaler by M.
2. The A down counter is loaded with an initial value of A at the rising edge of the M divider output and is clocked by the input signal of the M divider.
3. The A down counter value is reduced by one at every rising edge of its clock signal, except when the hold signal is high. When the down counter value reaches zero, it will remain zero unless the next load signal loads a start value to the counter.
4. The A down counter output is high when the counter value is nonzero, which toggles the dual-modulus divider to divide by $P + 1$, and its output is low when the counter value is zero, which toggles the dual-modulus divider to divide by P.
5. The hold input to the down counter can be connected to a fractional accumulator's carry out to achieve a fractional division ratio.

Note that the A counter can also be implemented as a modulo-A up counter. In this case, the reset pulse resets the counter to zero. The counter counts up at every clock edge and stops counting once the counter value reaches A. When the counter value is less than A, the modulus control output is high; when the counter value equals A, the modulus control output is low. Hence, the up counter can generate the same modulus control as the down counter described above. Ignoring the fractional accumulator for the moment, the hold signal of the down counter is always zero. When a division cycle begins, it is assumed that the prescaler starts dividing by $P + 1$ since the down counter value was reset to a nonzero value of A. The prescaler continues to divide by $P + 1$ for $(P + 1)A$ input cycles, after which the down counter's output changes state, and the prescaler starts to divide by P. Now, if $M \geq A$, there will be A clock cycles during which the down counter will count down from the initial value of A, to $A - 1$, $A - 2$, . . . , 1, providing a high output, during which time the dual-modulus divider divides by $P + 1$. For the remaining $M - A$ clock cycles (since $M \geq A$), the down counter value will remain at zero. This provides a low-level output during which time the dual-modulus divider divides by P. Counting the pulses at the input of the dual-modulus divider, there will be $(P + 1)A$ cycles. During this time, the dual-modulus divider is dividing by $(P + 1)$, and there will be $P(M - A)$ cycles during which the dual-modulus divider is dividing by P. This process ends when a rising edge is generated at the output of the M divider, and the down counter is reset to the initial value A. Thus, the average division ratio of the pulse-swallow divider is

$$\text{Div} = (P + 1)A + (M - A)P = PM + A \tag{6.26}$$

where the hold signal of the down counter was disabled, and $M \geq A$ was assumed. If $M < A$, it is evident that the dual-modulus divider will continue to divide by $P + 1$ (i.e., working only as a fixed-ratio divider) since the down counter never reaches zero between the load signals. Thus, the normal operation condition for the pulse-swallow divider with a dual-modulus prescaler is that $M \geq A$ or A can only be an integer number between 0 and M.

Often it is desirable to program the synthesizer continuously without skipping any possible channels. However, it is not always possible to satisfy this requirement for a pulse-swallow divider. To help understand this limitation in the use of a pulse-swallow divider, consider a 7/8 dual-modulus prescaler. Starting with $M = 1$ and $A = 0$ or 1 (since $M \geq A$), the pulse-swallow divider average division ratio is $PM + A = 7$ or 8. Similar analysis leads to a list of possible programming channels (divider ratios) as summarized in Table 6.16. As can be seen in the table, there are some channels that the synthesizer cannot program by programming divider ratios, when $M < P - 1 = 6$. No channels are skipped if $M > P - 2 = 5$, in which case some channels can be programmed in multiple ways (e.g., ratio 71 can be programmed by setting either $M = 9$, $A = 8$ or $M = 10$, $A = 1$). The only way to avoid overlap in channel programming is when $A < P = 7$. Thus, it is a waste of hardware to build a down counter that has a maximum size $A > P$. The programming restrictions for a pulse-swallow divider using a dual-modulus prescaler are summarized in Table 6.17. It is important to understand those restrictions in order to build a ΣΔ modulator to control this type of fractional-N architecture, since ΣΔ tends to modulate the divider ratio and dither it over a wide range. For this reason, one must be very careful with the required ratios that a pulse-swallow divider can program when a ΣΔ accumulator is used.

The lower frequency counters used in pulse-swallow dividers can be implemented using sequential digital logic circuits. A common implementation of a

Table 6.16 Possible Division Ratios of a Pulse-Swallow Divider Using a 7/8 Dual-Modulus Prescaler

M	A	$Div = MP + A$
Divider ratios are skipped when $M < P - 1 = 6$.		
1	0, 1	7, 8
2	0, 1, 2	14, 15, 16
3	0, 1, 2, 3	21, 22, 23, 24
4	0, 1, 2, 3, 4	28, 29, 30, 31, 32
5	0, 1, 2, 3, 4, 5	35, 36, 37, 38, 39, 40
No divider ratios are skipped when $M > P - 2 = 5$.		
No divider ratios overlap when $A < P = 7$.		
6	0, 1, 2, 3, 4, 5, 6	42, 43, 44, 45, 46, 47, 48
7	0, 1, 2, 3, 4, 5, 6, 7	49, 50, 51, 52, 53, 54, 55, 56
8	0, 1, 2, 3, 4, 5, 6, 7, 8	56, 57, 58, 59, 60, 61, 62, 63, 64
9	0, 1, 2, 3, 4, 5, 6, 7, 8, 9	63, 64, 65, 66, 67, 68, 69, 70, 71, 72
10	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10	70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80

Table 6.17 Pulse-Swallow Divider Programming Restrictions with a $P/P+1$ Dual-Modulus Prescaler

Average division ratio	$PM + A$
Normal operation condition	$M \geq A$, $A = 0 \sim \min\{P-1, M\}$
Condition without skipping channels	$M > P - 2$
Condition without overlapping channels	$A < P$
Minimum ratio for continuous programming	$P(P-1)$

modulo-A up counter is illustrated in Figure 6.28. The modulo-A up counter counts up at every input clock edge until it reaches a preloaded modulo value A represented by

$$A = 2^0 A_0 + 2^1 A_1 + 2^2 A_2 + \ldots + 2^{n-1} A_{n-1} \tag{6.27}$$

where A_0 through A_{n-1} are the modulo control bits. Note that this circuit uses toggle flip-flops (T-FF). Toggle flip-flops toggle their output when the input T is high and hold their output when T is low, that is, the next state $q^+ = \overline{q} \cdot T + \overline{T} \cdot q$. The clock signal of the toggle flip-flops is gated with the hold signal and modulus output signal. Thus, when the modulus signal is low or hold is high, the counter clock will be disabled, and the counter will stop counting. When the counter value reaches the modulo value A, the modulus control output will be set to low, and the feedback will stop the counting process until the next reset pulse comes.

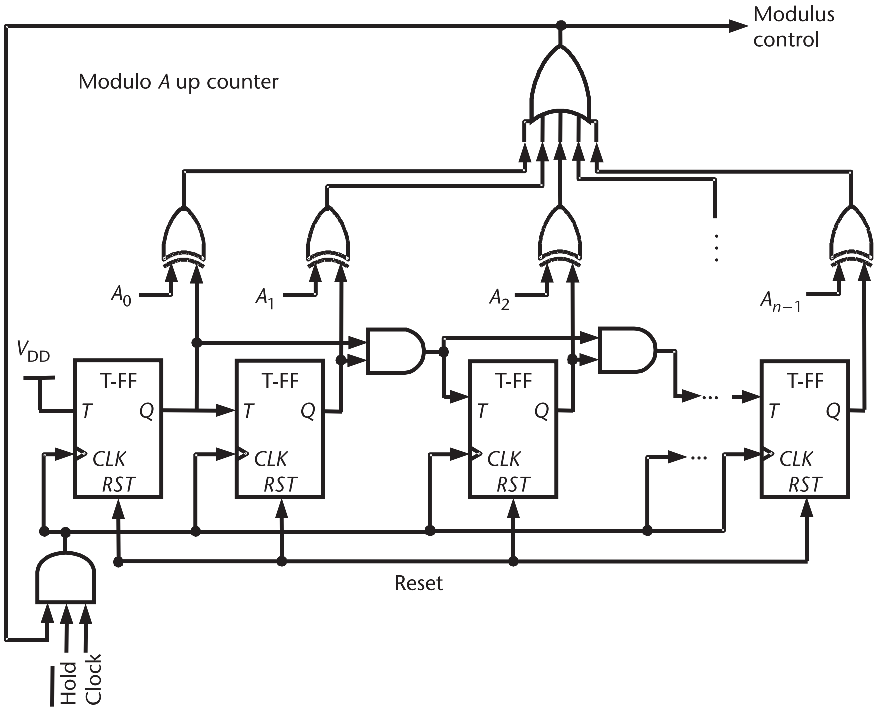


Figure 6.28 A modulo A up counter circuit.

The discussion so far has not shown how a fractional division ratio can be generated. The pulse-swallow divider technique discussed above is used to generate a programmable integer-divider ratio using a dual-modulus prescaler. With additional hardware, the pulse-swallow divider can be used to generate fractional-divider ratios as well. Figure 6.29 illustrates a fractional-N frequency synthesizer with a pulse-swallow divider. The presence of the hold signal for the down counter has thus far been ignored. The hold input of the modulo-A counter enables the circuit to take the fractional accumulator output and swallow additional input pulses at a fractional rate. If an accumulator is built whose carry-out is used as the hold signal for the swallow counter as shown in Figure 6.29, the counting process will be put on hold (the swallow counter value will not be reduced by one) for one cycle every time the accumulator carry-out is high. If the size of the accumulator is F with $\log_2 F$ as the number of bits, and the accumulator input word is K, then every F accumulator cycles, the carry-out will be high for K cycles. Thus, the average down counter value will be $A + K/F$. The architecture shown in Figure 6.29 has thus become a fractional-N synthesizer with dual-modulus prescaler. The synthesized frequency is given by

$$f_o = \frac{f_r}{R}\left(PM + A + \frac{K}{F}\right) \tag{6.28}$$

where f_r is the reference frequency provided by the crystal oscillator. K represents the frequency fine-tune word that controls the fractional part of the divider ratio,

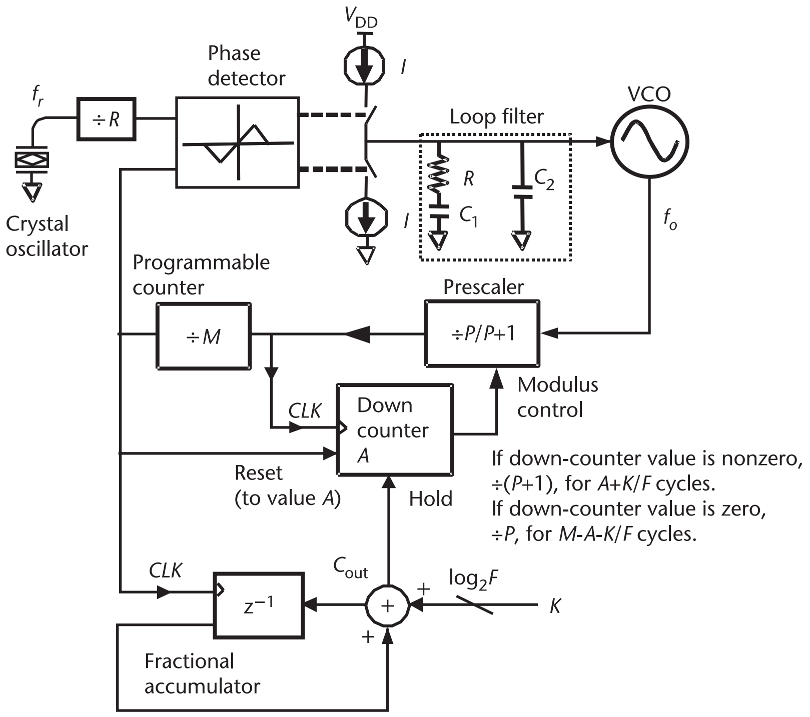


Figure 6.29 A fractional-N frequency synthesizer with a pulse-swallow divider.

while $PM + A$ provides a programmable coarse-tune word that controls the integer part of the divider ratio. This controllability makes it possible to keep the reference frequency high for low phase noise and to use the proper size of accumulator to get any desired step size.

6.3 Multipliers

Multiplier circuits are not as common as divider circuits, but sometimes it is necessary to multiply the frequency of the VCO. Often these circuits are not part of the loop but are used to drive mixers at different frequencies than the VCO is providing.

One way to build a multiply-by-two is to use an XOR gate. If an XOR gate is fed with two signals out of phase by 90°, then the output frequency will be at twice the input frequency ω_{in}. Such a circuit is shown in Figure 6.30. In this circuit, the input signal from the VCO is phase-shifted by highpass/lowpass RC filters that shift the phase by ±45°. The filter will work best if it is centered at the operating frequency of the multiplier. This means simply that

$$\omega_{\text{in}} = \frac{1}{RC} \tag{6.29}$$

In addition, it should be noted that because the RC phase shifter has a finite bandwidth, the circuit should be driven with sine waves. However, the sine waves should be of sufficient amplitude to switch the circuit completely to produce a square wave output.

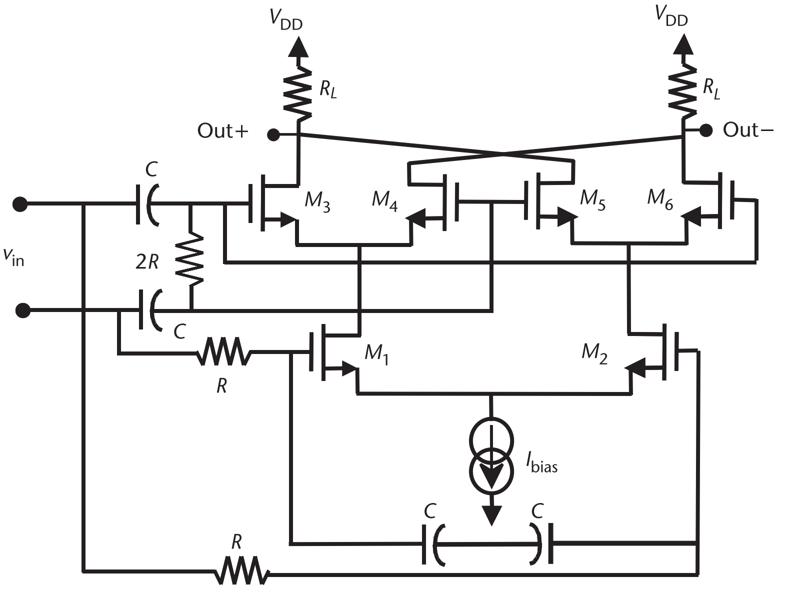


Figure 6.30 An XOR multiply-by-two circuit.

Another method of multiplying the frequency of a signal is to use a class C amplifier circuit with a tuned load centered at a multiple of the input frequency. Thus, a pulse of current from the transistors will excite the circuit once every N cycles of the output. Such a circuit, using bipolar transistors Q_1 and Q_2, is shown in Figure 6.31.

6.4 Phase Detectors

The following sections will deal with the implementation details of some common types of phase detectors. Although some of these circuits have already been mentioned in previous chapters, they will be considered more thoroughly here, including implementation details.

6.4.1 Basic Types of Phase Detectors

The XOR gate as a phase detector and the tristate phase detector have already been considered at the system level. The basic state diagram for the tristate phase detector as it is implemented is shown in Figure 6.32. Note that it actually has four states, but the fourth state is simply a reset state. Figure 6.32 shows that, in this state, which typically occurs once per cycle, both outputs are high simultaneously, although very briefly.

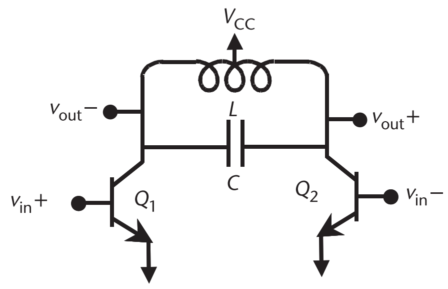


Figure 6.31 A class C–style multiplier.

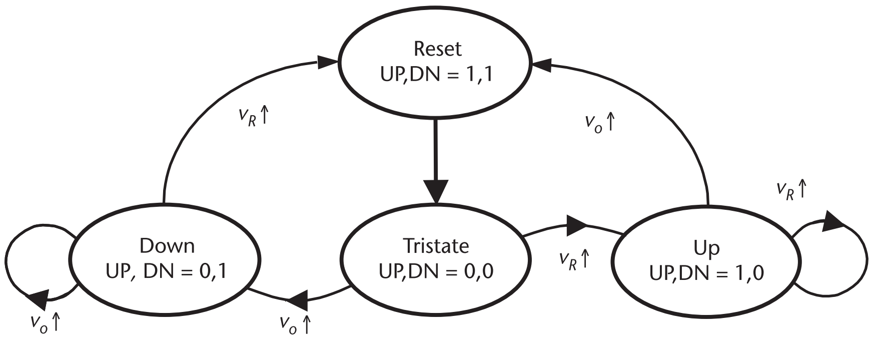


Figure 6.32 State diagram of a tristate PFD circuit.

The gain for this phase detector was given by

$$K_{\text{phase}} = \frac{I}{2\pi} \tag{6.30}$$

where I is the dc value of the current produced by the charge pump. The linearized curve of charge pump output current is shown in Figure 6.33. This circuit will experience cycle slipping each time the phase difference in the input and the VCO divider output exceeds 2π.

If cycle slipping is taking place, it is often assumed that the inputs are at different frequencies, so the PFD is operating as a frequency detector. In the transfer characteristics shown in Figure 6.34, the current remains positive for phase errors greater than 2π and negative for phase errors less than -2π to show operation as a frequency detector.

Even though the tristate phase detector is the most common phase detector in use today, there are many variations possible. Often cycle slip can be undesirable as it causes nonlinear behavior. By adding extra states to the tristate PFD, its linear range can be extended beyond 2π [6]. For instance, the PFD can be extended to have a range of 4π if two additional states are added, as shown in Figure 6.35.

In this circuit, there are four outputs from the PFD. They control four rather than two current sources. There are the two regular current sources that are usually present, plus two additional current sources used only during lock acquisition that provide additional current to the loop filter in order to accelerate the frequency-detection process. In principle, these two additional current sources can be even higher current than the regular ones for faster acquisition. Such a circuit is shown

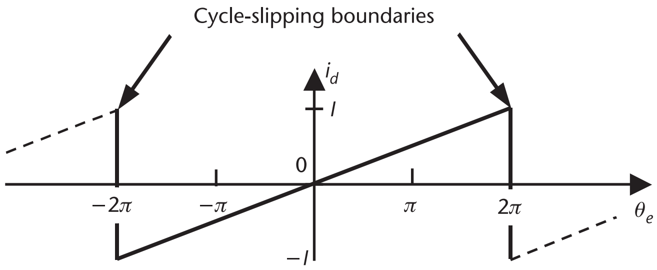


Figure 6.33 Linear transfer characteristic of the tristate PFD with charge pump.

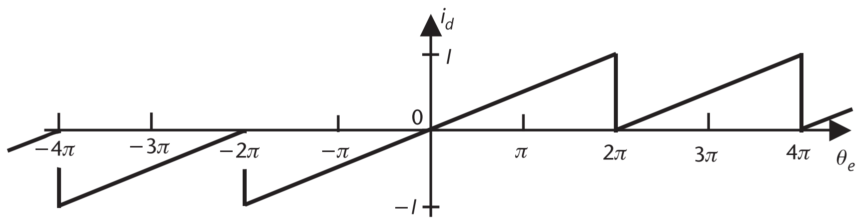


Figure 6.34 Tristate PFD with charge pump showing cycle-slipping and frequency-detector characteristics.

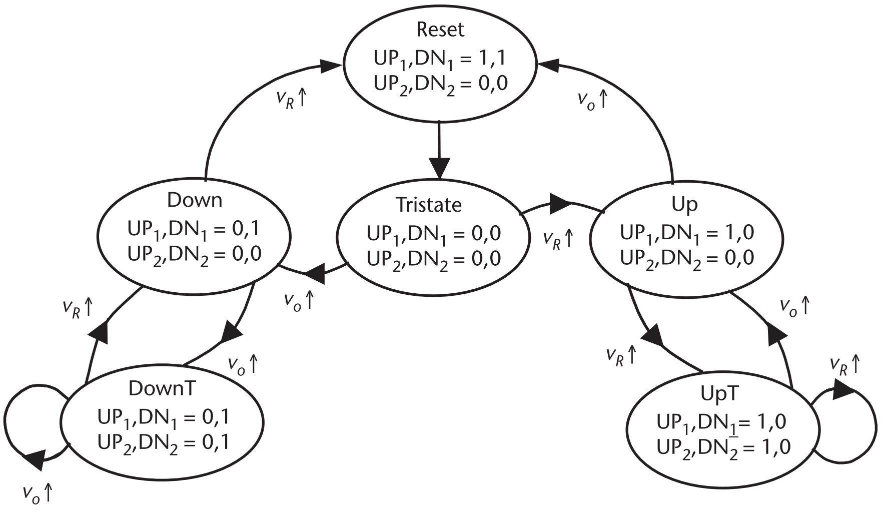


Figure 6.35 A five-state PFD circuit with additional linear range.

in Figure 6.36. Note that under normal (i.e., locked) conditions, the outer two states are not used, so the original two current sources are designed exactly as before. This circuit now has a range shown in Figure 6.37.

6.4.2 Circuit Implementations of PFDs

PFDs are usually constructed using basic digital cells. For instance, the tristate phase detector can be implemented as shown in Figure 6.38. Here the circuit is implemented as two resetable flip-flops and an AND gate. Both flip-flop inputs are tied high. The reference and the output of the divider are each fed into a clock input of one of the flip-flops. A rising edge of either the reference v_R or the output v_o immediately causes the corresponding flip-flop's output to go high. Once the other output is also high, the AND gate produces a one, which resets both flip-flops to the zero state until the next rising edge of either input arrives. In this way,

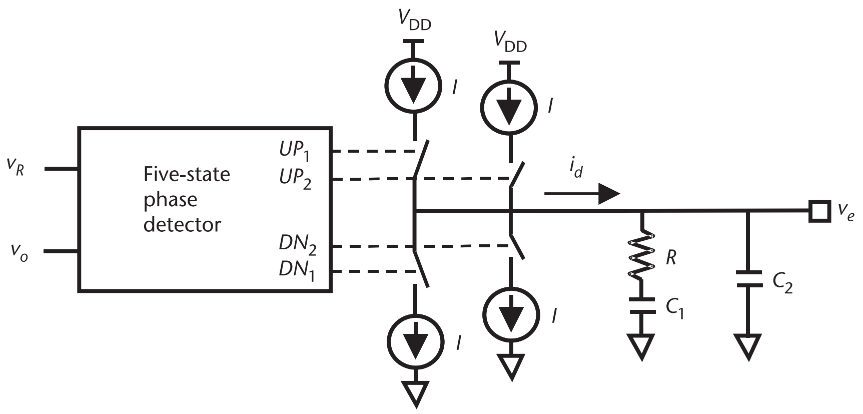


Figure 6.36 A five-state PFD circuit with additional linear range connected to a charge pump.

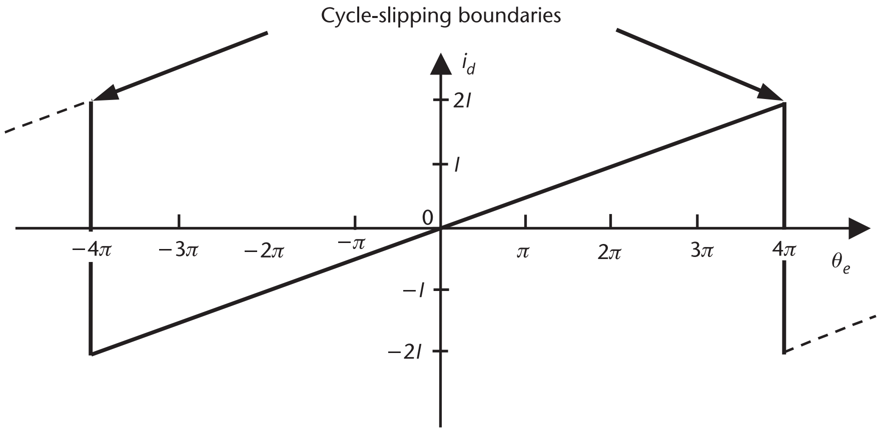


Figure 6.37 Linear transfer function of a five-state PFD circuit with additional linear range.

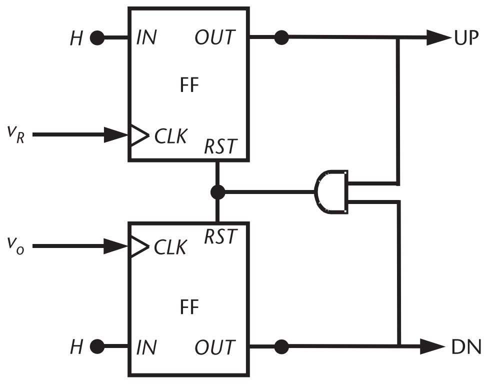


Figure 6.38 A tristate PFD circuit.

it should be easy to see that this simple circuit does implement the state diagram of Figure 6.32. The flip-flops themselves can be built with NOR gates, as shown in Figure 6.39.

In order to implement a five-state PFD, an additional nonstandard building block is required, as shown in Figure 6.40. This circuit consists of two delay blocks and a flip-flop. This circuit will produce a narrow pulse at its output one delay wide when the input goes high. This circuit is necessary to make sure that flip-flops in the five-state PFD are reset properly.

The five-state PFD itself could be implemented as shown in Figure 6.41. This circuit is built using a tristate PFD at its core with some added circuitry to extend it into a five-state machine. When either a reference v_R or an output v_o pulse arrives, an UP1 or DN1 pulse is triggered, if the PFD is not in either the UpT or DownT state. If the PFD is in the UpT or DownT state (also referred to as turbo states, where turbo is indicated by the "T" at the end of the state name), then the pulse will move the circuit out of the UpT or DownT state by resetting one of the two flip-flops on the right of Figure 6.41. If the circuit is not in either the DownT or the UpT state, then the arrival of a reference pulse clocks either *FF*1 and *FF*3,

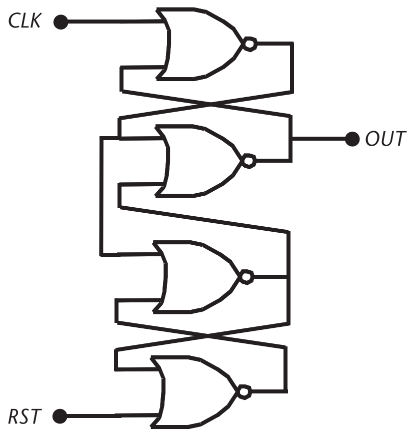


Figure 6.39 Positive edge-triggered *D*-flip-flop with active-low reset and hidden *D* = 1.

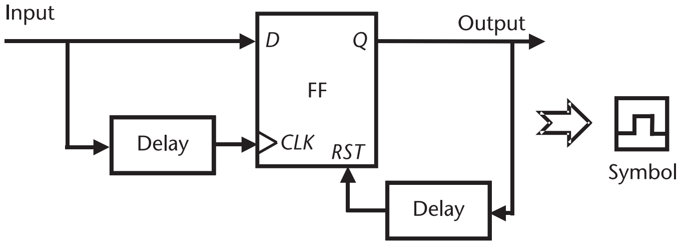


Figure 6.40 A narrow pulse generator.

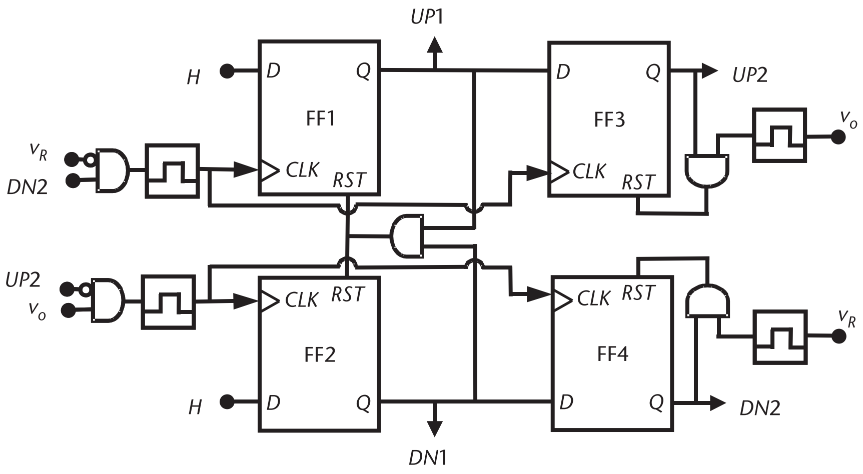


Figure 6.41 An implementation of a five-state PFD.

or *FF*2 and *FF*4. If the PFD was in the reset state, then *FF*1 or *FF*2 has its output go from a zero to a one, but since *FF*3 or *FF*4 would have had a zero at their input, then their output stays in the zero state, unless another pulse from the same input is the next event. In this case, *FF*3 or *FF*4 will have a one at the input, and the PFD will enter one of the two turbo modes. If a pulse arrives from the other input instead, then a reset action takes place, just as in the tristate circuit.

Example 6.3: Comparison of Five-State and Tristate PFD Settling Times
Using the loop parameters in Example 3.4, compare the settling times of the PLL design previously considered with the same design if the tristate PFD is replaced with a five-state PFD.

Solution: In this case, there is no redesign necessary. The tristate PFD is simply replaced with a five-state PFD, along with an additional charge pump. For the purposes of this example, the second charge pump is chosen to have the same current as the primary one. For a small step in frequency, there is, of course, no difference between the two loops since, if there is no cycle slipping, the second charge pump never becomes active. However, for a large step in frequency, the second charge pump does become active in the five-state design. The two loops are compared in Figure 6.42 for a 300-MHz frequency step at the output. The advantage of faster settling is evident in the five-state design.

6.4.3 Dead Zone in PFDs

For a small phase difference between the phase detector controlling signals, narrow pulses are required at the output. Due to finite rise and fall times, such narrow pulses cannot activate the charge pump, so the average output current will not follow the input phase. This region is called the dead zone and occurs primarily due to a difference in rise time between the latch outputs in Figure 6.38 and the reset path delay. To illustrate this point, suppose that the flip-flops in a PFD have a rise time of τ, as shown in Figure 6.43. Further, suppose that in order to turn on the charge pump, the output must reach a full logic level. Now, also suppose that the AND gate's threshold voltage is one-half of a logic level, and it resets the flip-flops much faster. The result is that the AND gate will reset the flip-flops at time $\tau/2$ after the second pulse starts to rise. If the pulses are nearly in phase, this means that the charge pump will never be turned on, as illustrated in Figure 6.44.

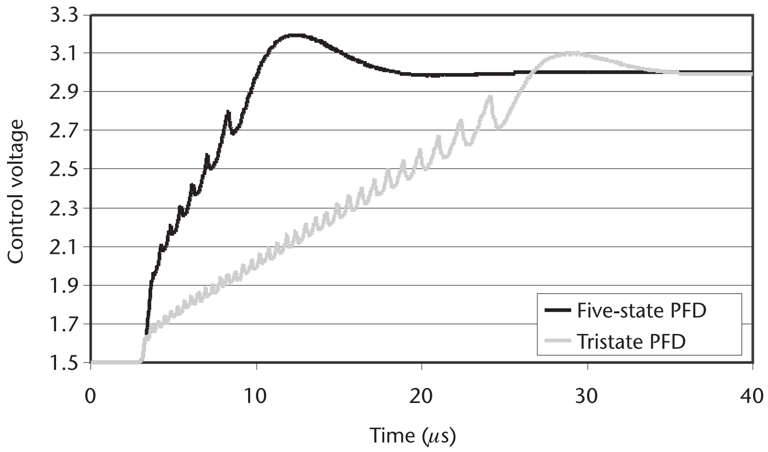


Figure 6.42 Comparison of a tristate and five-state PFD settling times.

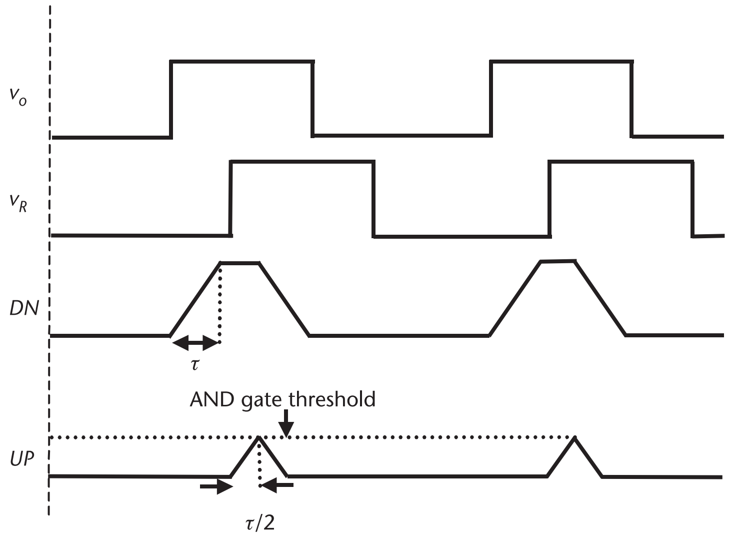


Figure 6.43 A PFD with finite rise time but a large difference in phase between inputs.

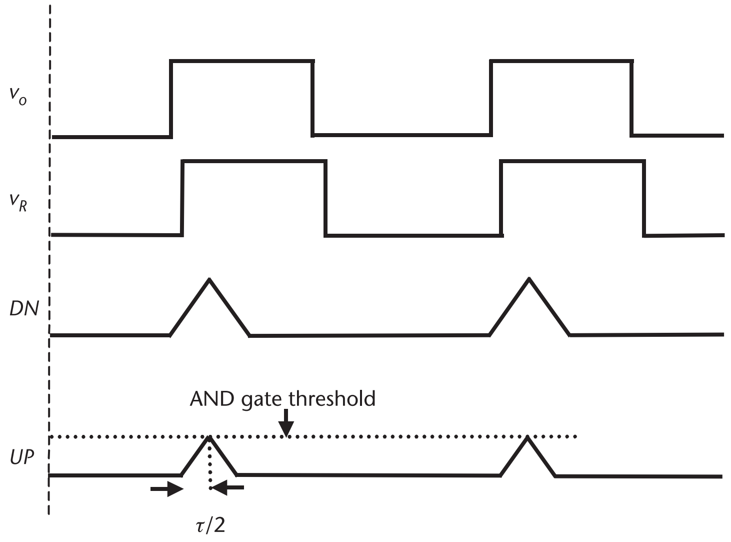


Figure 6.44 A PFD with finite rise time and a small difference in phase between inputs.

Thus, unless the time difference in the arrival of the pulses from the reference and the output is greater than $\tau/2$, the charge pump and PFD will remain inactive. This is called the dead zone, and for these small differences in phase, the loop is open, and no feedback will take place. Thus, the loop will only respond to differences in phase greater than

$$\text{Dead zone edge} = \pm \frac{\tau\pi}{T} \tag{6.31}$$

where T is the reference period. Thus, the dead zone will increase with a higher reference frequency or with an increased delay in the output.

A modified plot of the average output current as a function of the input phase difference is shown in Figure 6.45. One way to combat the dead zone is to add delay into the feedback path. If such delay is added to ensure that the time for reset is comparable to the delay in the forward path, then the dead-zone problem can be made less severe. For example, assume that delay is added in the feedback path so that the time for the AND gate to reset the flip-flops is also τ. Now assume that the phase difference between input and output is Δ, and that Δ is less than $\tau/2$. Then, with an equal delay in the reset path, the output will reach a one level, even for very small differences in phase, as shown in Figure 6.46. Adding further delay into the feedback path will cause both current sources to be on simultaneously, which is undesirable from a noise, spur, and power-consumption point of view, but does remove the dead-zone problem. Other examples of PFDs designed to remove the dead zone are given in [7, 8].

The presence of the dead zone means that, unless the phase difference between input and output reaches a certain finite value, the loop is essentially open. This means that phase noise that does not pass this threshold will not be suppressed by

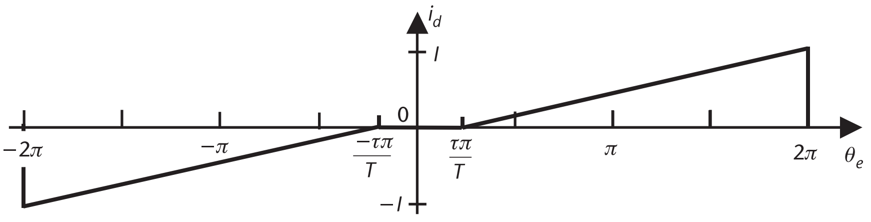


Figure 6.45 Plot of average output current of a PFD with a dead zone.

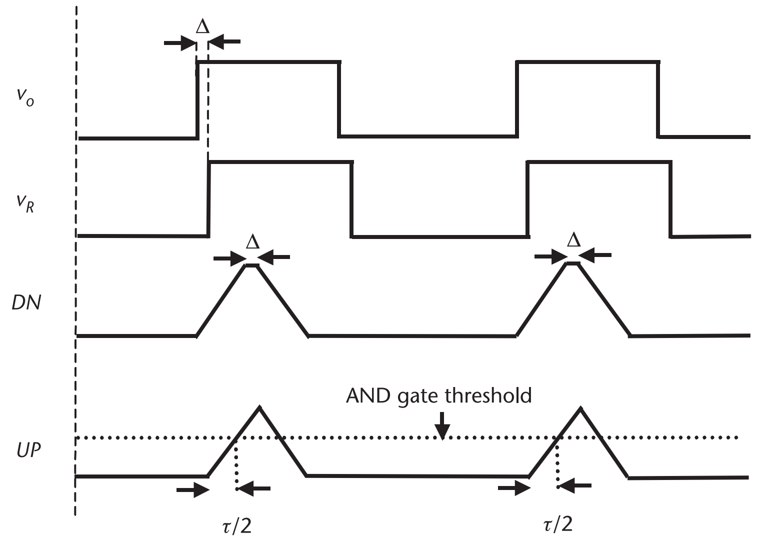


Figure 6.46 Plot of the output with delay inserted into the reset path.

the loop. Thus, for noise components very close to the carrier frequency, the loop is essentially open, and the phase noise of the output will be the raw phase noise of the VCO. For instance, assume that the VCO experiences an instantaneous impulse in its output frequency ω_o of value ω_{imp} such that the waveform coming out of the divider has the form

$$v_o(t) = A \sin\{[\omega_o + \omega_{\text{imp}}\delta(t)]t\} \tag{6.32}$$

This is compared with the reference that will be assumed to be ideal:

$$v_{\text{r}}(t) = A \sin(\omega_o t) \tag{6.33}$$

Now phase is the integral of frequency, so

$$\theta_o(t) = \int \omega_o + \omega_{\text{step}}\delta(t)\, dt = \omega_o t + \omega_{\text{imp}} \quad t > 0 \tag{6.34}$$

$$\theta_o(t) = \int \omega_o + \omega_{\text{step}}\delta(t)\, dt = \omega_o t \quad t < 0$$

for the signal that experiences the impulse of frequency change. Thus, the phase error (compared to the ideal reference, which has a phase of $\omega_o t$) is zero before the delta function event, and after it, the phase difference is ω_{imp}. As expected, a large jump in frequency will generate a large phase difference, but a very small jump in frequency will generate a very small change in the phase. Therefore, small jumps in frequency (which can be thought of as close in phase noise if noise is the source of the frequency jump) cannot get over the dead-zone threshold to activate the loop. In fact, the frequency jump must be

$$\omega_{\text{imp}} \geq \frac{\tau\pi}{T} \tag{6.35}$$

or the effective K_{phase} of the system will be zero, there will be no feedback, and the highpass transfer function that suppresses the VCO phase noise will fail, causing the close in phase noise (below the dead zone) to be equal to the raw VCO noise, as illustrated in Figure 6.47.

6.4.4 Lock-Detection Circuits

It is often very useful to have a digital signal that detects when lock has been achieved. This can be done for most of the PFDs considered. Note that, before lock is achieved, the signals coming from the PFD are changing state, but when lock is achieved, the PFD spends most of its time in the tristate. Thus, if a sensor, attached to the up and down outputs, puts out a lock-acquired signal when both these signals are in the same state, it is possible to tell when the circuit is in lock. Such a circuit can be as simple as an XNOR gate attached to UP and DN, as shown in Figure 6.48. In this circuit, when the PFD is not in a locked state, the

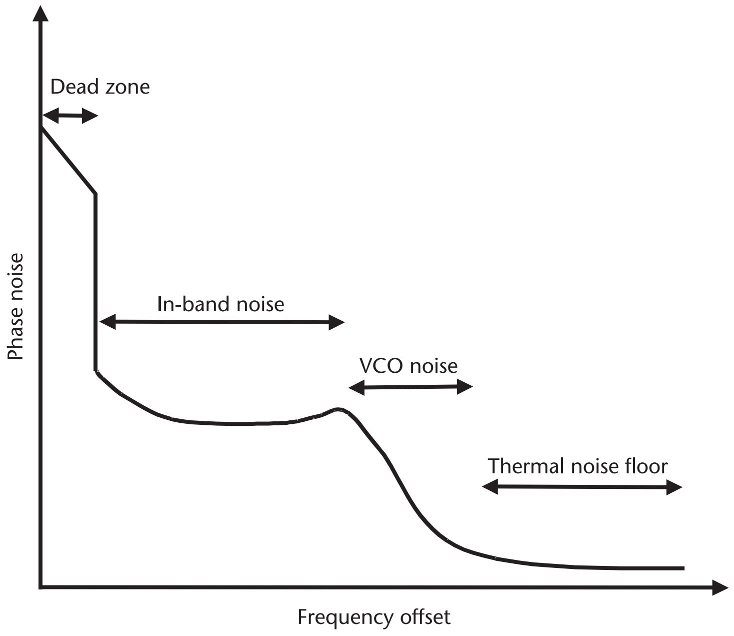


Figure 6.47 Illustration of the effect of dead zone on phase noise.

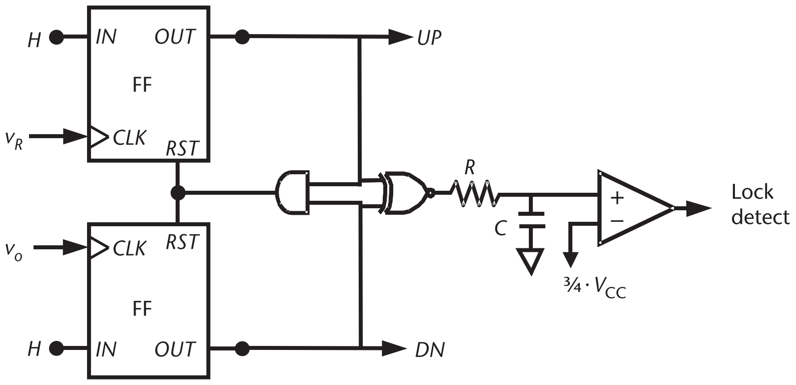


Figure 6.48 Tristate PFD with lock detection added.

dc output from the XNOR gate will be roughly half way between the power-supply rails, but when in lock, the XNOR gate will have a dc voltage that is almost V_{CC}. Thus, the dc output can be used to determine when the circuit is in lock. If a full rail output is desired, the output from the XNOR gate could be passed through a lowpass filter to clean up any high-frequency glitches and then through a comparator to make the signal compatible with rail-to-rail logic.

6.4.5 A Modified PFD with Aligned UP and DN Pulses

A modified version of the tristate PFD that is often used in practice is shown in Figure 6.49 [9]. This circuit behaves much like the regular tristate PFD, but when

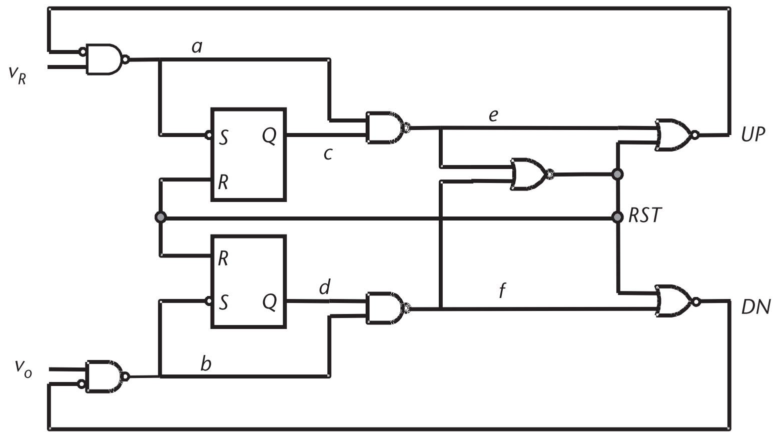


Figure 6.49 Modified tristate PFD.

an UP or DN pulse must also generate a reset pulse, the reset signal is generated directly without having to pass through a flip-flop. The reset pulse itself turns off both the UP and DN gates, bringing them low at exactly the same time. Otherwise, without such synchronization, timing error in the feedback could cause the flip-flops to turn off at slightly different times, causing reference feedthrough. In other words, a net charge would be dumped onto the capacitor, which would then have to be compensated for in the next period. This means that pulses of current would appear each cycle at the reference frequency. This would cause spurs and would cause the charge pump to produce more noise. Thus, this circuit helps to prevent this problem.

A detailed plot of the waveforms produced by the circuit in Figure 6.49 is shown in Figure 6.50. Note here that this PFD is falling-edge, rather than rising-edge, sensitive; other than that, it behaves much like a standard PFD. In the figure, the reference is leading the VCO, so the PFD produces an UP pulse. Note that on the rising edge of either clock input (connected to v_R and v_o), the flip-flop is set a full half-cycle before the PFD is required to produce an output. Once set, the flip-flop is ready for the falling edge of the clock. At the falling edge, the input signal passes through the second gate and changes the value of e. This creates an UP pulse. The UP pulse lasts until the falling edge of the VCO pulse, which causes f to go low. Once e and f are both low, a reset pulse is generated, which immediately pulls the two outputs (UP and DN) low again and simultaneously resets the flip-flops, making them ready for the next rising edge. Thus, the feedback path does not affect the timing of pulling both outputs low, as that is accomplished directly by the reset signal itself.

6.4.6 PFDs for CDR Applications

CDR circuits are used to generate a local clock from an incoming bit stream [10–12]. This local clock will then be used as timing for the incoming bits. Most of the components, such as charge pumps, loop filters, and VCOs, are similar to those used in the PLLs we have already been considering, except for two major

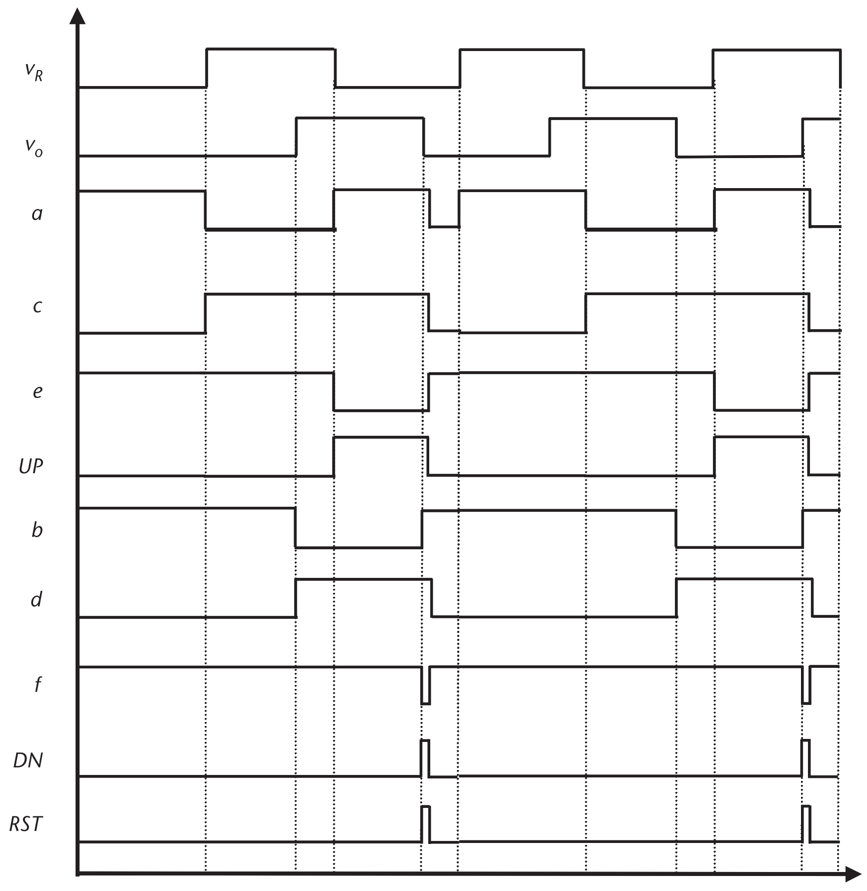


Figure 6.50 PFD voltage waveforms.

differences (and many small differences designed to fine-tune or tweak the performance). The first difference is that the input is at full rate; hence, there are no dividers in the loop. This means that the phase detector must be able to operate at full rate. The other important difference is that incoming bits can have a large number of zeros in a row or a large number of ones in a row. In either case, there are no transitions. In a conventional frequency synthesizer, the input is a reference signal, which does not miss any transitions. With the conventional PFD, with no incoming transitions, the output of the PFD would continue to see the VCO pulses, but no reference pulses and would therefore produce a DN signal. The DN signal would cause the output of the charge pump to head to the rails. In a CDR system, phase detectors need to be designed to include a transition detector such that, when there are no input pulses, the circuit still functions correctly.

6.4.6.1 The Hogge Phase Detector

The Hogge phase detector, illustrated in Figure 6.51, is a simple circuit comprising two flip-flops and two XOR gates. The UP and DN signals in this circuit control

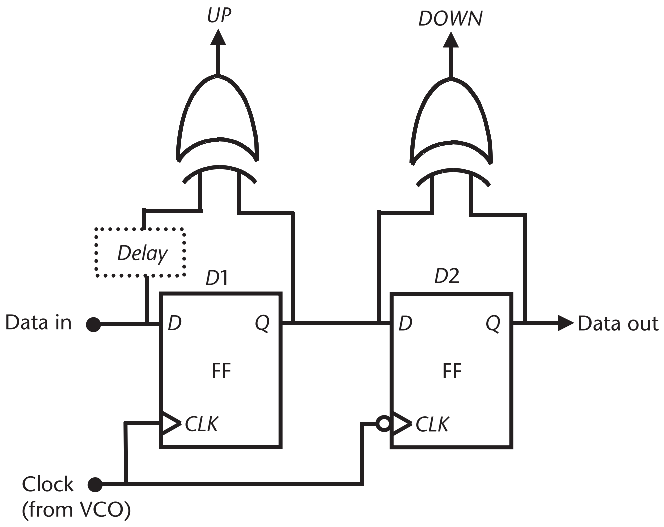


Figure 6.51 The Hogge phase detector.

a charge pump just as they would in the case of a regular PFD [13]. Without a transition in the bit stream, the inputs of the XOR gates will be the same and the output will not change. When there is a transition in the data, *Data In* and the output of *D*1 will be different until the next clock edge, as shown in Figure 6.52. Thus, assuming the delay in the dashed box is not present, the UP signal will be active for

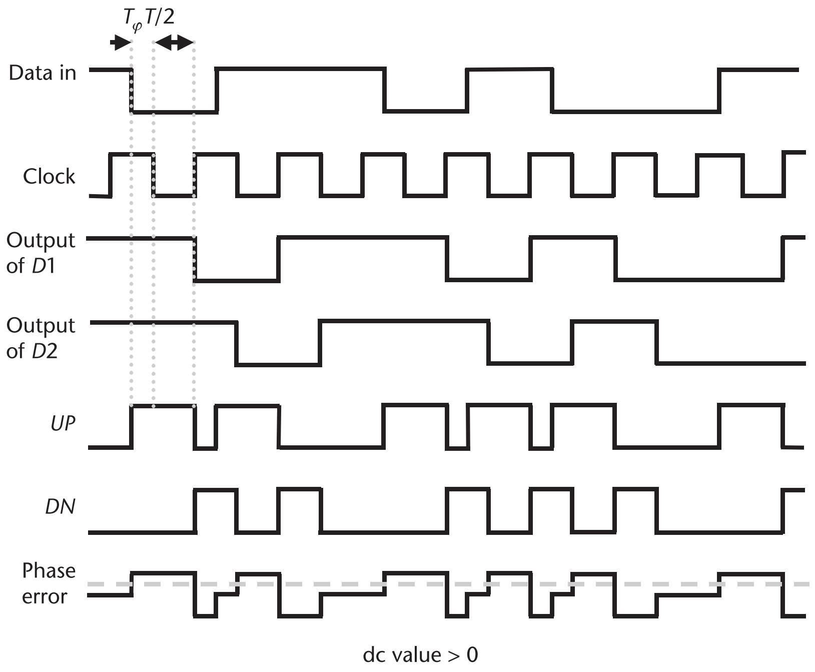


Figure 6.52 Timing diagram for the Hogge phase detector with the input data ahead of the clock.

$$T_{\mathrm{UP}} = \frac{T}{2} + T_{\varphi} + T_{\text{clk-to-q}} \tag{6.36}$$

where T_{UP} is the time the UP pulse is active, T is the clock period, T_{φ} is the time equivalent to the phase difference between the data and VCO; and $T_{\text{clk-to-q}}$ is the delay through the flip-flop (not shown in Figure 6.53). Now the change in *Data In* will be passed onto the second flip-flop $D2$. Thus, its input and output will be different for half a clock cycle:

$$T_{\mathrm{DOWN}} = \frac{T}{2} \tag{6.37}$$

where T_{DOWN} is the time the DN output is active. Note that the input has already been synchronized with the clock so that this output is high for exactly half a clock period. Also note that, assuming the delays in both flip-flops are equal, the output will be high for exactly half a clock period, as the delay in the input will be the same as the delay in the output. Using this phase detector, the charge pump will experience an UP and DN signal for each bit transition that will be different in length by T_{φ} (ignoring flip-flop delay). The charge pump will inject a net charge into the loop filter until the phase difference is corrected. Thus, the average current dumped onto the loop filter over the cycle is

$$I_{\mathrm{AVE}} = \frac{T_{\varphi}}{T} I = \frac{\varphi}{2\pi} I \tag{6.38}$$

Therefore, for the Hogge phase detector the gain is again given by

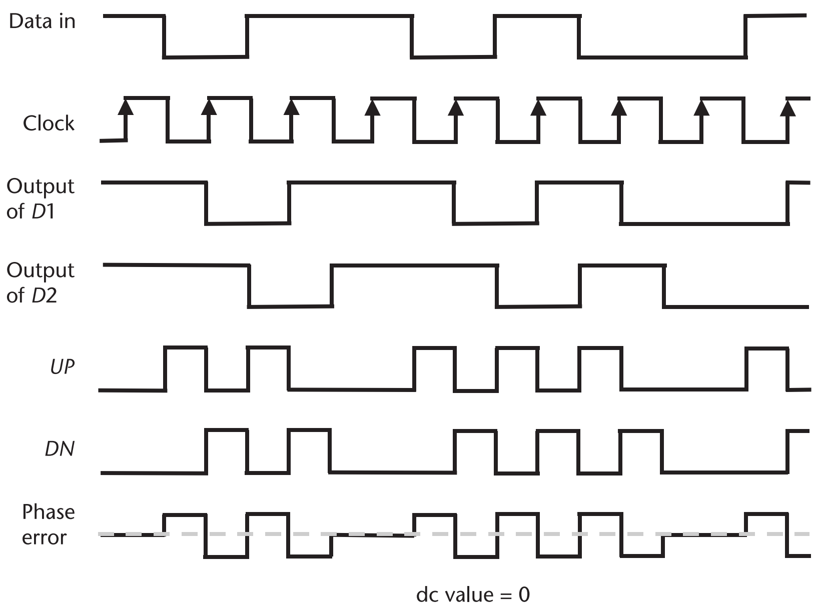


Figure 6.53 Timing diagram for the Hogge phase detector with the clock and data optimally aligned.

$$K_{\text{phase}} = \frac{I}{2\pi} \tag{6.39}$$

Note that, even in the locked condition, as shown in Figure 6.53, the charge pump will have its current sources on a lot of the time. One way to get around this problem is to delay the UP pulse by half a clock period, thus reducing ripple on the control line. Also note that in order to get rid of the problem with flip-flop delay causing a mismatch in pulse width, a delay equal to one flip-flop delay can be added to the circuit, as shown in Figure 6.51.

6.4.6.2 The Bang-Bang Phase Detector

There are other types of phase detectors and CDR loops in existence, notably, the so-called Bang-Bang PLL. The phase detector often used with this architecture is the Alexander phase detector [14], as shown in Figure 6.54. Like the Hogge phase detector, this circuit only generates outputs when there is a data transition.

In this circuit, when there are no data transitions, then all flip-flops have the same value, and there is no output. The case for which a data transition does occur is illustrated in Figure 6.55. In this figure, the first data pulse arrives before the clock. As a result, the next rising and falling clock edges change the value of both $D1$ and $D2$. Then, on the next clock edge, these 2 bits are passed onto $D3$ and $D4$ simultaneously because the clock is late; thus, the DN signal remains low. Now, because the clock is late, on the second rising edge after the data has arrived, a zero is clocked into $D1$. $D1$ and $D4$ having different values indicates that the clock is late; therefore, they activate the UP signal. Similarly, when the clock is early, $D2$ will go high first. This will cause $D4$ to go high a clock cycle before $D3$. Thus, a DN pulse will be generated as expected. In the case when the clock is on time, flip-flop $D1$ will be in a meta-stable state. Its input will change at the same time that it gets a rising clock edge. Thus, it will be a toss-up as to whether

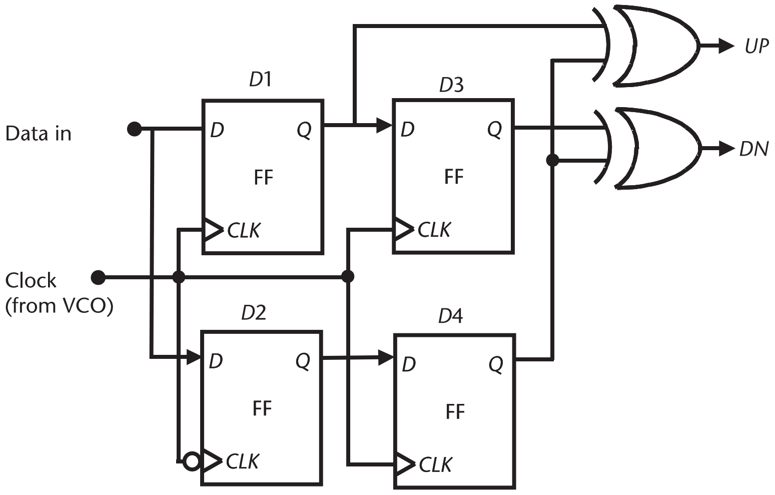


Figure 6.54 The Alexander phase detector.

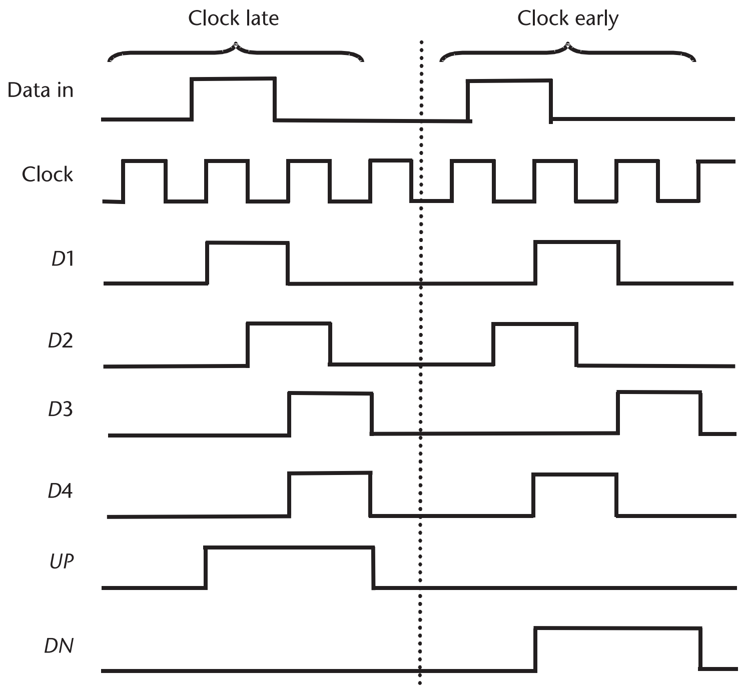


Figure 6.55 Waveforms in the Alexander phase detector.

a zero or a one is passed to the output. One of these two events will happen and then, during the next clock cycle, the loop will act to correct the phase imbalance. Thus, the phase detector will have infinite gain in the locked condition ($K_{\text{phase}} = \infty$) and will tend to "bang" around the correct phase.

References

[1] Magoon, R., and A. Molnar, "RF Local Oscillator Path for GSM Direct Conversion Transceiver with True 50% Duty Cycle Divide by Three and Active Third Harmonic Cancellation," *IEEE Radio Frequency Integrated Circuit Symposium*, Seattle, WA, 2002, pp. 23–26.

[2] Vaucher, C. S., et al., "A Family of Low-Power Truly Modular Programmable Dividers in Standard 0.35 μm CMOS Technology," *IEEE J. Solid-State Circuits*, Vol. 35, July 2000, pp. 1039–1045.

[3] Vaucher, C. S., and D. Kasperkovitz, "A Wide-Band Tuning System for Fully Integrated Satellite Receivers," *IEEE J. Solid-State Circuits*, Vol. 33, July 1998, pp. 987–997.

[4] Sandireddy, R. K. K. R., F. F. Dai, and R. C. Jaeger, "A Generic Architecture for Multi-Modulus Dividers in Low-Power and High-Speed Frequency Synthesis," *IEEE 5th Topical Meeting on Silicon Monolithic Integrated Circuits in RF Systems*, Atlanta, GA, September 2004. pp. 243–246.

[5] Craninckx, J., and M. S. J. Steyaert, "A 1.75-GHz/3-V Dual-Modulus Divide-by-128/129 Prescaler in 0.7-μm CMOS," *IEEE J. Solid-State Circuits*, Vol. 31, July 1996, pp. 890–897.

[6] Wolaver, D. H., *Phase-Locked Loop Circuit Design*, Upper Saddle River, NJ: Prentice Hall, 1991.

[7] Craninckx, J., and M. S. J. Steyaert, "A Fully Integrated CMOS DCS-1800 Frequency Synthesizer," *IEEE J. Solid-State Circuits*, Vol. 33, December 1998, pp. 2054–2065.

[8] Ultra High Speed Phase/Frequency Discriminator, AD9901 1999, Analog Devices product data sheet, 1999, http://www.analog.com/en/prod/0,2877,AD9901,00.html.

[9] 3.3V/5V ECL Differential Phase-Frequency Detector, MC100EP40/D June 2002, on semiconductor product data sheet, January 2004, http://www.onsemi.com/pub/Collateral/MC100EP40-D.PDF.

[10] Razavi, B., *Monolithic Phase-Locked Loops and Clock Recovery Circuits*, New York: Wiley-IEEE Press, 1996.

[11] Razavi, B., *Design of Integrated Circuits for Optical Communications*, New York: McGraw-Hill, 2002.

[12] Greshishchev, Y. M., and P. Schvan, "SiGe Clock and Data Recovery IC with Linear-Type PLL for 10-Gb/s SONET Application," *IEEE J. Solid-State Circuits*, Vol. 35, September 2000, pp. 1353–1359.

[13] Hogge, C. R., "A Self-Correcting Clock Recovery Circuit," *IEEE J. Lightwave Tech.*, Vol. 3, December 1985, pp. 1312–1314.

[14] Alexander, J. D. H., "Clock Recovery from Random Binary Data," *Electronic Letters*, Vol. 11, October 1975, pp. 541–542.

CHAPTER 7

Charge Pumps and Loop Filters

7.1 Introduction

A charge pump is the first analog component of a synthesizer that will be considered. It is responsible for placing charge into or taking charge out of the loop filter, and, therefore, moving the control voltage on the VCO up or down, as shown in Figure 7.1. Charge pumps must be very carefully designed to minimize reference feedthrough or phase noise. In spite of numerous variations, a charge pump will usually be composed of two current sources and two switches. This chapter also deals with loop filters. Although Chapter 3 has already considered basic loop filters, a large number of modifications to improve performance can be added to the basic design.

7.2 Charge Pumps

The following sections will deal with various charge pump design issues, and circuit-level designs will be presented.

7.2.1 A Basic Charge Pump

Probably one of the most basic and common charge pumps is shown in Figure 7.2. This charge pump consists of four transistors M_1 through M_4. Transistors

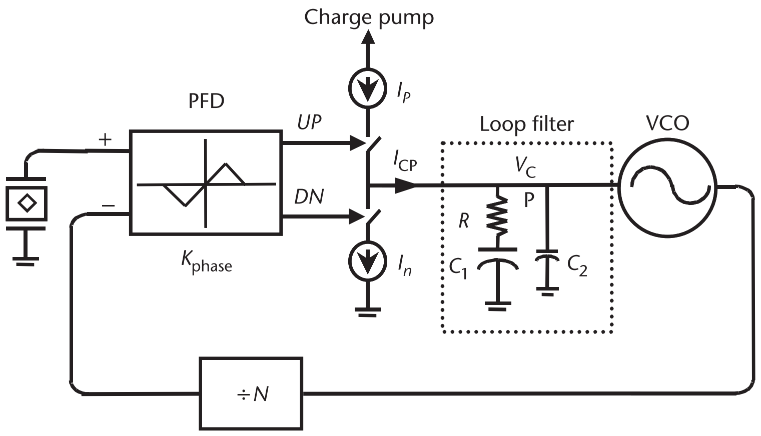


Figure 7.1 A system-level diagram of a basic synthesizer.

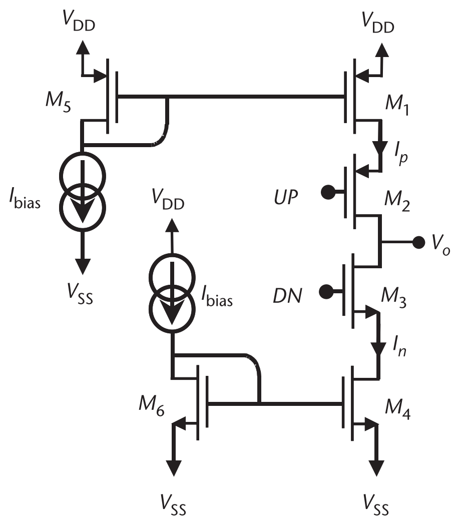


Figure 7.2 A basic, single-ended charge pump circuit.

M_1 and M_4 act as current sources, while transistors M_2 and M_3 act like switches that are controlled by the PFD. When the PFD is in the tristate, then UP must be high (at V_{DD}), while DN must be connected low (to V_{SS}). Note that an inverter between the PFD output and the DN or UP input of the charge pump may be required. In order to ensure that the switches turn on and off when the charge pump output voltage is close to the power-supply rails, the signal from the PFD must be a full rail-to-rail signal. As a result, interfacing this particular charge pump with a CML-based PFD is difficult as a rail-to-rail output buffer would be needed.

Later sections will suggest improvements for this charge pump design; however, most of them will still contain the basic current sources M_1 and M_4. In the following sections, a number of issues regarding transistor-sizing of M_1 and M_4 will be considered.

7.2.2 Saturation Voltage

The saturation voltage of M_1 and M_4 in Figure 7.2 is very important. The lower the saturation voltage on the current source transistors, the closer to the rail the charge pump is able to operate properly without changing the loop gain and, therefore, the loop dynamics. The drain source saturation voltage of a MOS transistor is given by

$$v_{DS\,Sat} = v_{GS} - V_T \tag{7.1}$$

where v_{GS} is the gate-source voltage, and V_T is the threshold voltage of the transistor. For a review of transistors and their parameters, see Appendix B. (Note that equations are given for NMOS devices. For PMOS, voltages are reversed, as described in Appendix B, or the absolute values of voltages are used; for example, the saturation voltage for a PMOS transistor is $|v_{DS\,Sat}| = |v_{GS}| - |V_T|$). The drain current in the saturation region is

$$i_D = \frac{1}{2}(\mu C_{\text{ox}})\left(\frac{W}{L}\right)(v_{GS} - V_T)^2 \tag{7.2}$$

where W is the transistor total width, L is the transistor length, μ is the electron mobility in silicon (or hole mobility in the case of PMOS), and C_{ox} is the gate capacitance per unit area. Therefore, (7.1) can be rewritten as

$$v_{\text{DS Sat}} = v_{GS} - V_T = \sqrt{\frac{2i_D}{\mu_n C_{\text{ox}}}\left(\frac{L}{W}\right)} \tag{7.3}$$

Therefore, to allow the VCO control voltage to get as close to the rails as possible, a large W/L ratio and a low drain current are preferred. In general, charge pump current transistors tend to be large. This is important because, if operation close to the supply rails is possible, the full VCO tuning range can be used.

7.2.3 Current Source Output Impedance

The second major problem with current source transistors in a charge pump is that they have finite output resistance. Even in the saturation region, their drain voltage will vary with the output voltage so that, even if the currents are matched at mid-rail, they will not be perfectly matched at all output voltages. Further current mismatch can occur due to mismatch of NMOS and PMOS transistors, for example in their output resistance, as shown in Figure 7.3, or due to temperature drift. To keep this difference as small as possible, it is desirable to keep the output resistance of the transistors as high as possible. The output resistance is given by

$$r_{DS} = \frac{1}{\lambda I_{DS}} \propto \frac{L}{I_{DS}} \tag{7.4}$$

where λ is the output impedance constant and is inversely proportional to channel length (see Appendix B). As the voltage moves away from mid-rail, if the output

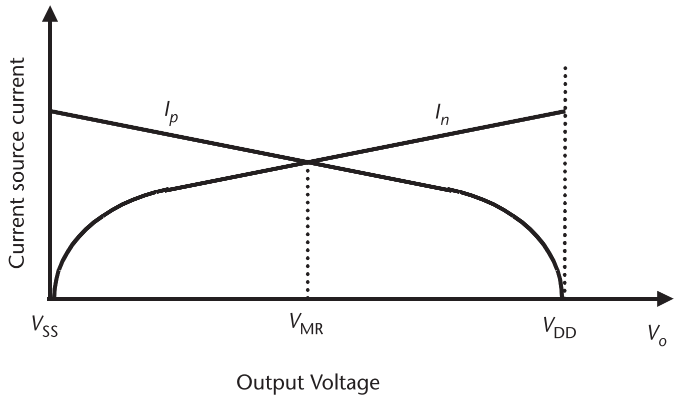


Figure 7.3 Current source output currents for a charge pump made with transistors that have a finite output impedance.

impedances are equal, then there will be a difference in the up and down currents given by

$$\Delta I = 2 \cdot \frac{(V_{oCP} - V_{MR})}{r_{DS}} \propto \frac{(V_{oCP} - V_{MR}) I_{DS}}{L} \tag{7.5}$$

where V_{oCP} is the voltage at the output of the charge pump and V_{MR} is the mid-rail voltage. Therefore,

$$\frac{\Delta I}{I_{DS}} \propto \frac{(V_{oCP} - V_{MR})}{L} \tag{7.6}$$

Thus, we see that, for the purposes of current matching, having a long device is good. Note that bipolar transistors have much higher output resistance, but the matching between npn and pnp transistors could potentially be much worse. Note, as described in Appendix B, that npn and pnp are bipolar transistors where n and p refer to n-doped and p-doped semiconductors.

There are many ways to increase the output resistance of current sources, but most are at the expense of headroom. One way to increase the output resistance is by adding resistive degeneration to the circuit, as shown in Figure 7.4(a). In this case, using Figure 7.4(b), the output resistance can be found to be

$$R_{out} = \frac{V_X}{I_X} \approx r_{DS}(1 + R_S g_{m4}) \tag{7.7}$$

where g_m is the transconductance of a transistor.

Another way to increase the output resistance of a current source is by adding a cascode transistor. The modified current source is shown in Figure 7.4(c). This circuit can be analyzed with the aid of Figure 7.4(d). In this case, there is no voltage feedback to the gate of M_4; therefore, the current source g_{m4} does not come into the equation. Thus, the total output resistance can be found as

$$R_{out} = \frac{V_X}{I_X} \approx r_{DS7}(1 + r_{DS4} g_{m7}) \tag{7.8}$$

Bipolar versions of these current sources end up having slightly different formulas due to the finite base current. A bipolar current source with degeneration resistance R_E, shown in Figure 7.5, has an output impedance of [1]:

$$R_{out} \approx \frac{R_E g_m r_o}{1 + \frac{R_E g_m}{\beta}} \tag{7.9}$$

where β is the ratio of the collector to base current. If R_E is increased or represents the output impedance of another transistor (cascaded bipolar), then, in the limit, if $R_E g_m \gg \beta$,

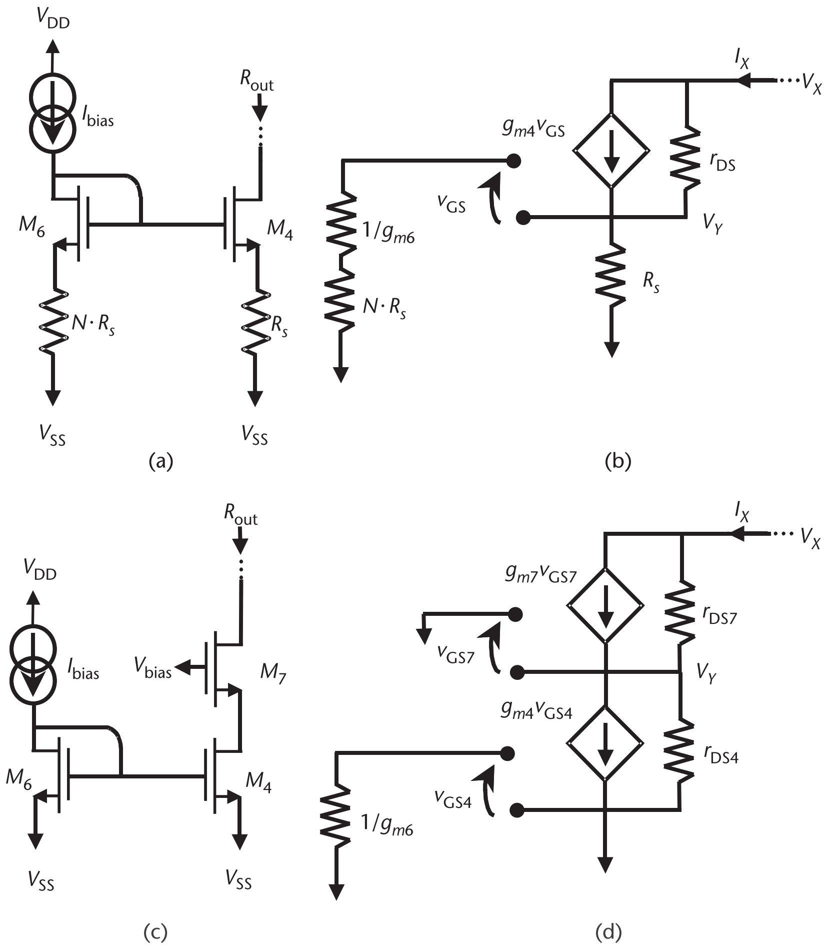


Figure 7.4 Current source with increased output resistance due to (a) degeneration, and (c) a cascode transistor. The associated small signal models are given in (b) and (d), respectively.

$$R_{\text{out}} \approx \beta r_o \tag{7.10}$$

This is the maximum value of the output impedance for a bipolar current source. CMOS, on the other hand, has no such limit. Since CMOS saturation voltages are also quite low, it can often be possible to stack more than two devices for even higher output impedance, as shown in Figure 7.6.

7.2.4 Reference Feedthrough

Reference feedthrough can affect how much time the charge pump must remain on in the locked state. If the phase difference between the reference and the feedback from the VCO is zero, then both the current sources are on for an instant while the PFD resets itself. If the two currents are mismatched in that instant, then a net charge will be deposited onto the loop filter, forcing a correction during the next cycle of the reference. This will create an ac signal on the control line of the VCO

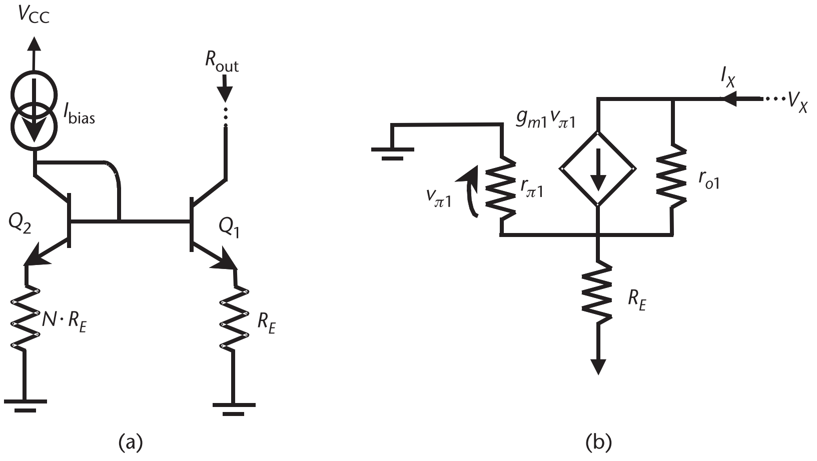


Figure 7.5 (a) Bipolar current source with increased output resistance due to degeneration resistors, and (b) approximate small-signal model for computing the output impedance.

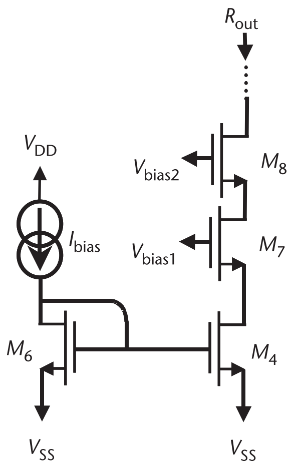


Figure 7.6 Current source with increased output resistance due to multiple cascode transistors.

at the reference frequency, as shown in Figure 7.7. If the current sources are mismatched by an amount ΔI, and the reset path has a delay of time δ, then a charge q is placed on the loop filter, where q is of value

$$q = \delta \cdot \Delta I \tag{7.11}$$

This will require the other current source to be on for a time t to remove the charge:

$$t = \frac{\delta \cdot \Delta I}{I_{CP}} \tag{7.12}$$

Thus, the total time that the charge pump will be on over one cycle will be

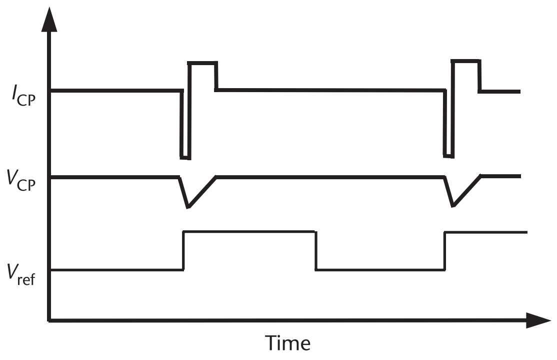


Figure 7.7 Graph showing the effect of mismatched current in a charge pump.

$$t_{CP} = \delta\left(1 + \frac{\Delta I}{I_{CP}}\right) \tag{7.13}$$

Now, since these current pulses happen at the reference frequency and well beyond the corner frequency of the loop, and if a typical loop filter is assumed, such as the one discussed in Chapter 3, then all this current will flow into C_2 of Figure 7.1. Thus, the voltage will have a triangular shape and a peak amplitude of

$$V_{mm} = \frac{\delta \cdot \Delta I}{C_2} \tag{7.14}$$

From basic Fourier series analysis, this triangle wave will have a fundamental component with an amplitude of

$$V_{ref} = \frac{\delta^2 \cdot \Delta I}{C_2 T_0}\left(1 + \frac{\Delta I}{I_{CP}}\right) \tag{7.15}$$

provided that the current flows for a small fraction of a period T_0. This signal on the control line of the VCO will modulate the output of the VCO according to the formula

$$v_{out}(t) \approx A\cos(\omega_o t) + \frac{A V_{ref} K_{VCO}}{2\omega_{ref}}\left[\cos(\omega_o + \omega_{ref})t - \cos(\omega_o - \omega_{ref})t\right] \tag{7.16}$$

Thus, the magnitude of the reference spurs relative to the carrier will be

$$\text{Spurs} = 20\log\left[\frac{\delta^2 \cdot \Delta I \cdot K_{VCO}}{4\pi \cdot C_2}\left(1 + \frac{\Delta I}{I_{CP}}\right)\right] \text{dBc} \tag{7.17}$$

Example 7.1: Reference Feedthrough Due to Charge Pump Mismatch

Starting with the loop in Example 3.4, if the UP current source is 10% high, what reference feedthrough will result if the reset delay in the PFD/CP is assumed to be 5 ns?

Solution: In Example 3.4, values of note were C_2 = 566 pF, I_{CP} = 628 μA, and K_{VCO} = 200 MHz/V. In this case, the UP current is assumed to be 691 μA; therefore, ΔI = 63 μA. The spurs can be predicted from (7.17):

$$\text{Spurs} = 20\log\left[\frac{\delta^2 \cdot \Delta I \cdot K_{VCO}}{4\pi \cdot C_2}\left(1 + \frac{\Delta I}{I_{CP}}\right)\right]$$

$$= 20\log\left[\frac{(5\text{ ns})^2 \cdot 63\ \mu\text{A} \cdot \left(2\pi \cdot 200\dfrac{\text{MHz}}{\text{V}}\right)}{4\pi \cdot 566\text{ pF}}\left(1 + \frac{63\ \mu\text{A}}{628\ \mu\text{A}}\right)\right]$$

$$= -71\text{ dBc}$$

This can also be simulated. A result of a fast Fourier transform (FFT) of the output of the synthesizer with a 10% mismatch in charge pump currents is shown in Figure 7.8. In this plot, simulations show that the spurs are at a level of –69.5 dBc, which is very close to what the theory above predicted.

7.2.5 Transistor Gain Considerations

While charge pumps in many applications operate at low frequencies relative to the speed available with most modern integrated technologies, it would be easy to think that speed is not a concern when designing a charge pump. This, however, is not true. The problem is that, when the charge pump current sources turn on and off, they do so in a finite amount of time; therefore, current mismatch will be worsened if the up and down current sources do not turn on and off at the same rate. Thus, it is best to match the transconductance of both the NMOS and PMOS current sources. The transconductance is given by

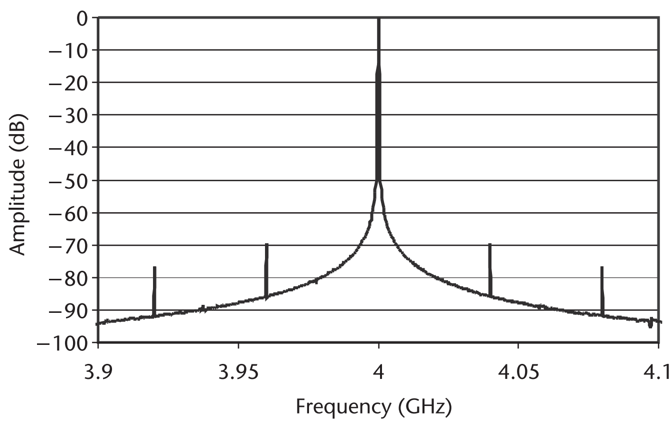


Figure 7.8 Simulated synthesizer output spectrum with charge pump current mismatch of 10%.

$$g_m = \sqrt{2\mu C_{\text{ox}}\left(\frac{W}{L}\right) I_{\text{DS}}} \tag{7.18}$$

where μ is the mobility of the transistor in question. Since μ is typically two to three times larger in an NMOS transistor than in a PMOS transistor, the *W/L* ratio of the PMOS must be two to three times higher than the *W/L* ratio for an NMOS transistor. This is accomplished by increasing either the number of fingers in the PMOS by two to three times or the multiplicity of the PMOS by two to three times. If the *W/L* ratio is to be adjusted, then the length of all devices should be the same for both the NMOS and PMOS devices, and only the width should be scaled. Adding fingers can also be done, and if it is, then it is best to keep the number of fingers odd for matching reasons. To improve matching further, it is best to scale the number of transistors, and, if possible, to use a common centroid layout [2].

The speed of the circuit will affect the dead zone of the PFD/CP system. Even if the PFD is designed to be dead-zone free, if the charge pump itself cannot respond quickly enough, then this can also result in reduced gain at low phase differences. The charge pump can also create nonlinearities in K_{phase}, thus changing the loop gain to unexpected values.

7.2.6 Charge Pump Noise

Noise is another major issue with charge pumps. Since, in band, charge pump noise can often dominate the noise of the system, it is very important to keep charge pump noise as low as possible. We need to be aware of two major sources of noise in a MOSFET. They are the drain noise (channel noise) and flicker noise, as shown in Figure 7.9. The first is the major source of white noise in the device, and it is usually assigned a value of

$$i_{\text{nd}}^2 = 4kT\left(\frac{2}{3}\right) g_m \tag{7.19}$$

We can rewrite this expression as

$$i_{\text{nd}}^2 = 4kT\left(\frac{2}{3}\right)\sqrt{2\mu C_{\text{ox}}\left(\frac{W}{L}\right) I_{\text{DS}}} \tag{7.20}$$

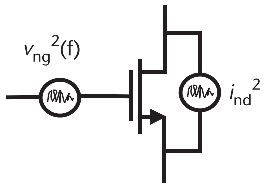


Figure 7.9 Noise sources in a MOSFET.

Therefore, to have low output thermal noise, we would like the device to have a low g_m, which generally means not wide and as long as possible. Low width is at odds, though, with many of the other requirements previously discussed.

The other major source of noise in a MOSFET is flicker noise. The flicker noise is inversely proportional to frequency; therefore, it dominates at lower frequencies. The gate-referred flicker noise is given by

$$v_{\text{ng}}^2(f) = \frac{K}{WLC_{\text{ox}} f\alpha} \tag{7.21}$$

where α is approximately 1 and K is a constant of proportionality that is process dependent. Note that in most processes, it is common for PMOS transistors of equivalent size to have lower $1/f$ noise than NMOS transistors.

Now, if we assume that the source is grounded, we can refer the flicker noise to the output:

$$i_{\text{ng}}^2(f) = \frac{K}{WLC_{\text{ox}} f} g_m^2 = \frac{K}{WLC_{\text{ox}} f} 2\mu C_{\text{ox}}\left(\frac{W}{L}\right) I_{\text{DS}} = \frac{2K\mu}{L^2 f} I_{\text{DS}} \tag{7.22}$$

Thus, for low output flicker noise, we would like to operate the device at low current and make sure that the channel is very long.

Phase noise is of interest mainly in the locked state and is of less importance when acquiring lock. If the charge pump is well designed and has well-matched up and down currents, the charge pump will spend the majority of its time in the off state when the loop is locked. This means that the noise actually transferred to the output will be relatively small.

The total noise produced by one current mirror while it is on is

$$i_{\text{no total}}^2(f) = \frac{2K\mu}{L^2 f} I_{\text{DS}} + 4kT\left(\frac{2}{3}\right)\sqrt{2\mu C_{\text{ox}}\left(\frac{W}{L}\right) I_{\text{DS}}} \tag{7.23}$$

Since we have two current sources that are on for some length of time t_{CP} for each period, the total noise produced by the charge pump on average is

$$i_{\text{no total\_average}}^2(f) = 2 i_{\text{no total}}^2(f) \frac{t_{\text{CP}}}{T_0} \tag{7.24}$$

Thus, the noise performance of a charge pump is related not only to the transistor noise sources but also to a number of other parameters that determine t_{CP}.

Note that what is of the most concern is the charge pump's effect on the phase noise of the PLL into which it is placed. Thus, it is the phase noise contribution of the charge pump that should be minimized, not its output noise current. The phase noise of the charge pump is referred to the PLL input by dividing (7.24) by the gain of the PFD/CP. Therefore, the input-referred total phase noise will be

$$\theta_n(f) = \frac{\sqrt{2 i_{\text{no total}}^2(f) \frac{t_{\text{CP}}}{T_0}}}{K_{\text{phase}}} \tag{7.25}$$

for a tristate phase detector. Given the above, the simple model for noise in a charge pump becomes

$$\theta_n(f) = 2\pi \sqrt{\left[\frac{2K\mu}{L^2 f I_{\text{DS}}} + 4kT\left(\frac{2}{3}\right)\sqrt{\frac{2\mu C_{\text{ox}} W}{I_{\text{DS}}^3 L}}\right]\left(\frac{t_{\text{CP}}}{T_0}\right)} \tag{7.26}$$

Thus, for better phase noise performance from a charge pump, higher current is better. Of course, if increased current comes at the price of higher on time, this may limit the obtainable phase noise improvement.

7.2.7 Charge Sharing

Another problem with the charge pump, as we have shown in Figure 7.2, is charge sharing [3]. Suppose that there is capacitance connected to the drains of M_1 and M_4. When the switching transistors are open, the voltages on these two nodes move towards V_{DD} and V_{SS}, respectively. When the switches close, instantaneously, some of the charge stored on these parasitic capacitors will be transferred to the loop filter. This is known as charge sharing. This can cause voltage spikes on the VCO control line, which can cause spurs in the output spectrum of the synthesizer. This is not desirable, so, in order to remove these spurs, it is necessary to keep the voltage on these nodes equal to the output voltage of the circuit. Charge sharing, along with the presence of current mismatch, can lead to reference feedthrough due to transients on the control line of the VCO. Improved circuits to reduce reference feedthrough will be considered in the next sections.

7.2.8 Improving Matching Between I_p and I_n

This section discusses the detailed design of the current mirrors. An important consideration is the matching of the current sources I_p and I_n. The simpleminded approach would be to connect the two currents to a single bandgap current, as shown in Figure 7.10. Provided that the transistors are all properly matched in size, and provided that the output impedance is infinite, the up and down currents will be perfectly matched. However, in practice, perfect matching between I_p and I_{bias} will only be achieved when the drain voltage of M_1 is equal to the drain voltage of M_5. Similarly, perfect matching between I_n and I_{bias} will be achieved when the drain voltage of M_4 equals the drain voltage of M_7. If it is assumed that the on resistance of the switches M_2 and M_3 is zero, then these two conditions will be met for the same output voltage $V_o = v_{G5}$; therefore, at this voltage, $I_p = I_n$. If the output voltage is higher than this value, then the current I_p will be lower (due to reduced current from r_{DS1}), and the current through r_{DS4} will be higher, causing the total current I_n to be higher than I_p. If the voltage goes lower than v_{G5}, I_p will be higher than I_n.

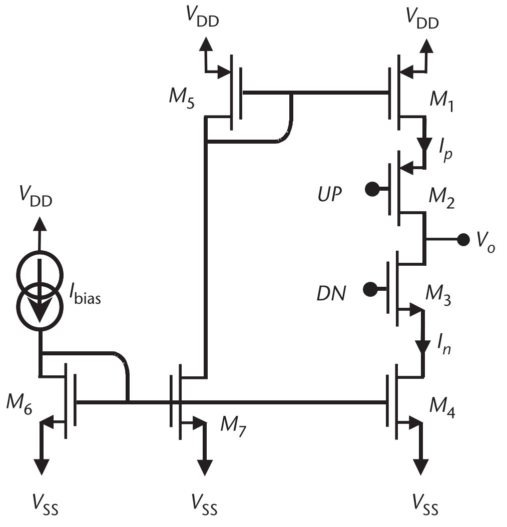


Figure 7.10 Charge pump with biasing shown in more detail.

Ideally, the currents I_p and I_n should be exactly equal, not only when the output happens to be at v_{G5} but for any output voltage. This can be accomplished by introducing feedback into the biasing scheme with a single op-amp, as shown in Figure 7.11 [4]. The op-amp senses the output voltage and compares it with the voltage at the drains of the mirror transistors. If the voltage on the mirrors is higher than the voltage at the output, then it increases the voltage on the gate of M_5 and M_1. This causes the current through M_5 to decrease slightly, forcing the currents through both the NMOS and PMOS transistors to be almost exactly equal, regardless of the output voltage. Note that, for stability, the polarity of the op-amp is important. This circuit, as shown, has both a negative and a positive feedback

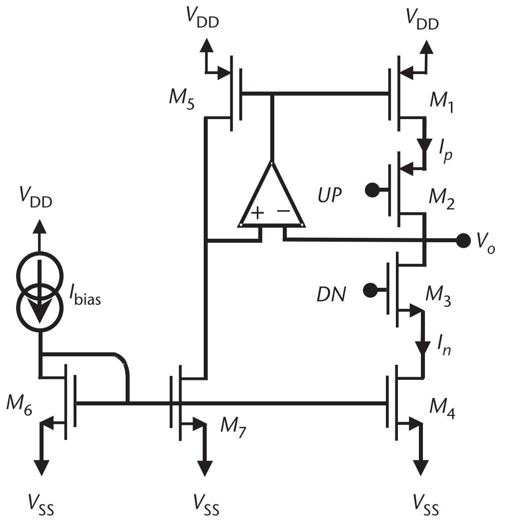


Figure 7.11 Charge pump with biasing shown with feedback to equalize UP and DN currents.

loop; however, in steady-state operation, the switch is open most of the time, also opening the positive feedback loop. Also note that there will still be a finite error in this circuit's currents due to the finite gain of any real op-amp and finite switch resistance. However, this circuit is now much improved over previous implementations.

Another improvement can be made to this basic charge pump. The required headroom of this circuit can be decreased by moving the switches out of the signal path and onto the control lines as shown in Figure 7.12.

7.2.9 Charge Pumps Compatible with CML/ECL

The charge pump circuits discussed so far are compatible with rail-to-rail CMOS logic but are less appropriate when used with CML. A much better topology to use with a CML PFD is shown in Figure 7.13 [5, 6]. In this circuit, the input differential pairs (Q_1 and Q_2 or Q_{11} and Q_{12}) steer current either to a dummy load (R_{B10} and R_{B5}), when both input signals (UP and DN) are low, or into the two current source transistors (Q_6 and Q_7), when the input signals are high. Note that, with the DN signal, there is an extra current direction reversal because the current must be pulled downwards. Bipolar transistors are often a superior choice to CMOS transistors as charge pump current sources because of their very low flicker noise. However, in an integrated environment, the gain of the pnp transistor is usually quite low. In addition, since pnp transistors have a very low beta, current matching in this circuit may be much more of an issue than it would be with a CMOS charge pump.

A CMOS version of this charge pump is shown in Figure 7.14. This is identical to the bipolar version except that, as is common with CMOS circuits, there are

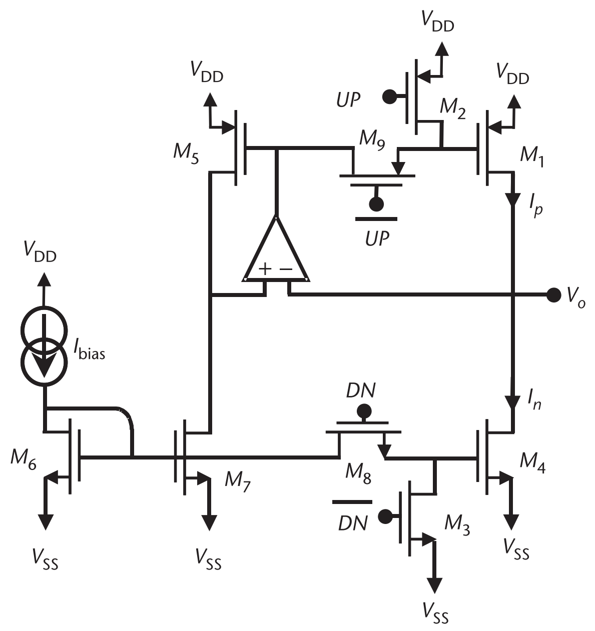


Figure 7.12 Charge pump with bias feedback and switches moved out of the signal path.

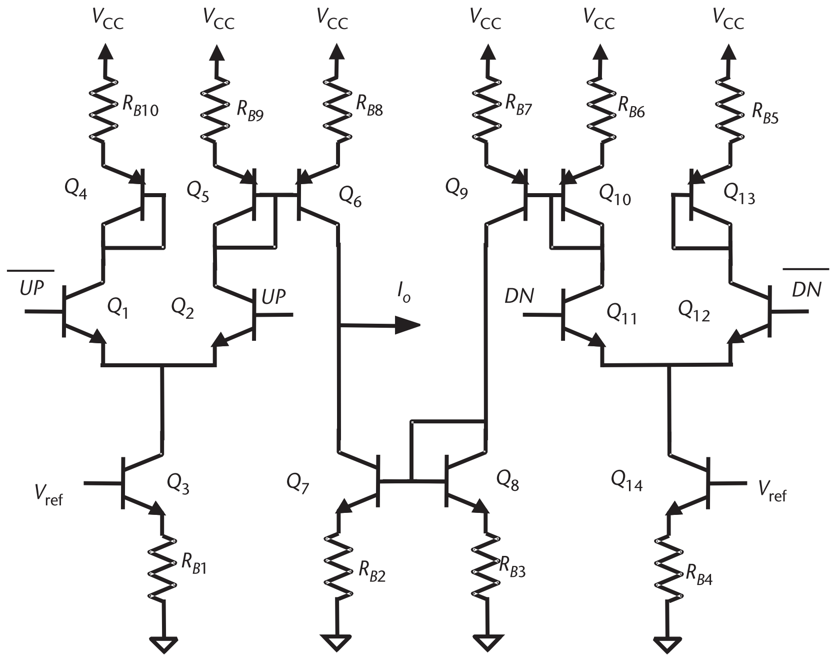


Figure 7.13 A bipolar charge pump that works well with a CML-based PFD.

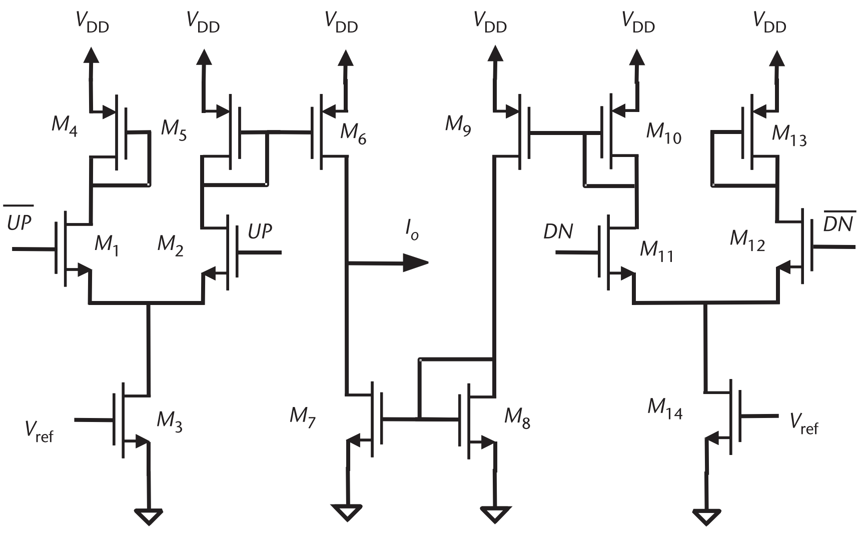


Figure 7.14 A CMOS charge pump that works well with a CML-based PFD.

no degeneration resistors used with this circuit. In a Bi-CMOS technology, the input differential pairs may still be made with bipolar transistors for ease of switching, even if the charge pump itself is CMOS.

Charge pumps that are compatible with CML are not quite as power efficient as the previous charge pumps shown because there is always some dc current flowing through some of the branches of the circuit. These circuits in Figures 7.13

and 7.14 will have good current matching because the UP and DN input stages are very symmetric.

Example 7.2: Charge Pump Design

Design a charge pump based on the circuit topology shown in Figure 7.14. Using a typical 0.18-μm process, size all transistors and determine the current for best phase noise performance. Assume a 3-V supply, and make sure that the loop control voltage is able to come within 350 mV of either rail. The charge pump is to be used in a PLL with a 40-MHz reference. Assume the PFD has a feedback delay of 1 ns for the purposes of this example.

Solution: Charge pump design can require careful simulation due to the nonlinear switching of the circuit. The design of CML circuits is covered in Chapter 5, including the proper sizing of transistors M_1, M_2, M_3, M_{11}, M_{12}, and M_{14}, and will not be discussed in detail here. Now the main challenge will be to determine the optimal current for this design. Whatever current is chosen, a saturation voltage of about 300 mV or less will need to be maintained. Thus, the W/L ratio will have to follow the relationship of

$$\frac{W}{L} = \frac{2i_D}{v_{\text{DS Sat}}^2 \mu C_{\text{ox}}}$$

Since $v_{\text{DS Sat}}$ must remain constant, the W/L ratio will be directly proportional to the current in the charge pump. Now, as the current increases, from (7.26), it can be expected that the input-referred noise performance will improve, but, at the same time, the capacitance in the circuit will rise, so the on time of the circuit will also increase. The matching in the circuit will also start to suffer if the current is made too large, causing more noise, but this will not be considered here. For low $1/f$ noise contribution, a length of 2 μm will be chosen for all transistors. Now the width W of the main charge pump transistors M_6 and M_7 will be determined by the above equation, but the scaling of the mirrors needs to be determined. At first, it might seem advantageous to scale M_5 and M_8 to save current; however, this can result in increased noise. For example, if M_8 is scaled to be N times smaller than M_7, the equivalent circuit model is shown in Figure 7.15. In this case, the equivalent output current noise generated by the mirror transistor is

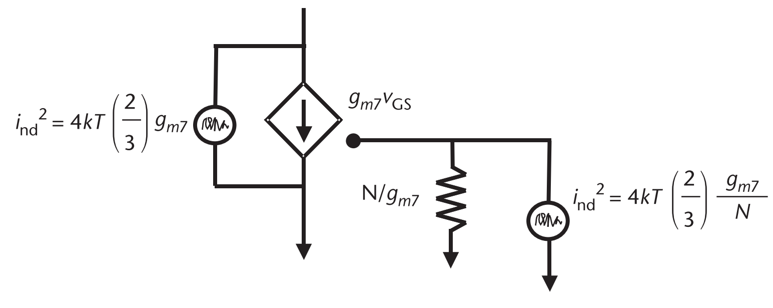

$i_{nd}^2 = 4kT\left(\frac{2}{3}\right)g_{m7}$
$g_{m7}v_{GS}$
N/g_{m7}
$i_{nd}^2 = 4kT\left(\frac{2}{3}\right)\frac{g_{m7}}{N}$

Figure 7.15 Illustration of noise in mirrors inside a charge pump.

$$i_{\mathrm{no}}^2 = 4kT\left(\frac{2}{3}\right) g_{m7} N$$

Thus, the noise current will increase as the mirror ratio gets larger. Therefore, a unity current ratio is preferred for noise.

Output noise is then simulated for currents ranging from 0.5 to 4 mA. To accommodate these currents, the width of the NMOS current source ranges from 100 to 700 μm. For g_m matching, the PMOS transistor in this process must be 4.8 times larger than the NMOS transistor. In order to test the charge pump for noise, a dc voltage source is attached to the output with a 1Ω resistor to measure the noise voltage and current. The two inputs (UP and DN) are driven with a low-duty-cycle square wave, and the output noise is measured with a nonlinear simulation, such as Spectre's PSS/pnoise analysis from Cadence Inc. The output noise currents are shown in Figure 7.16.

It is no surprise that the output noise is proportional to the charge pump current. However, the most important result is the input-referred phase noise, which is obtained by dividing the output noise current by $I/2\pi$, the phase detector gain. This normalized plot is shown in Figure 7.17. It can be seen that, for all these simulations, 2.5 mA is optimal for best noise performance. So why is there an optimum current in this design? As the current is increased, the size of the transistors must also increase, resulting in increased on time due to increased capacitance in the circuit. At some point, the on time grows faster than the linear-current-to-noise ratio. Thus, there is an optimum current for a charge pump in a given technology. Note that the current found here is a little on the high side due to the exaggerated PFD feedback delay. In a real design with a 40-MHz reference clock, the current would most likely be between 1 and 2 mA for best performance. However, in all cases, careful simulation must be performed to find the exact number.

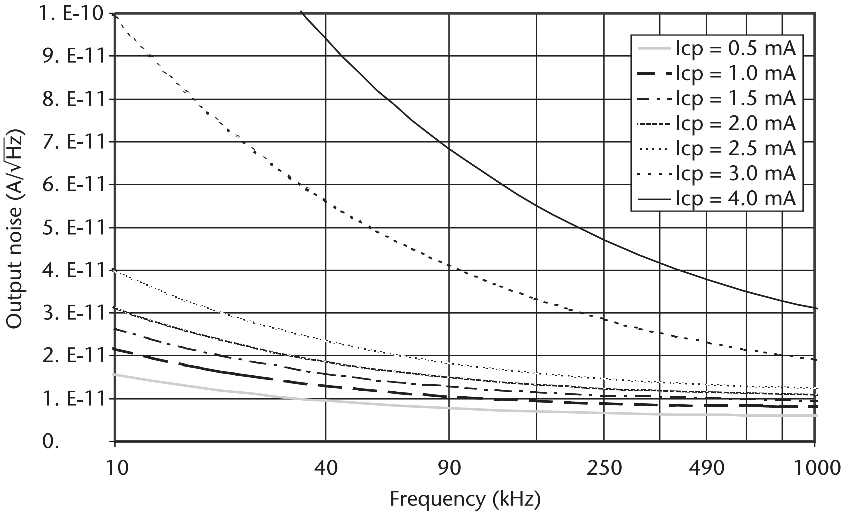


Figure 7.16 Output noise for various currents in the charge pump.

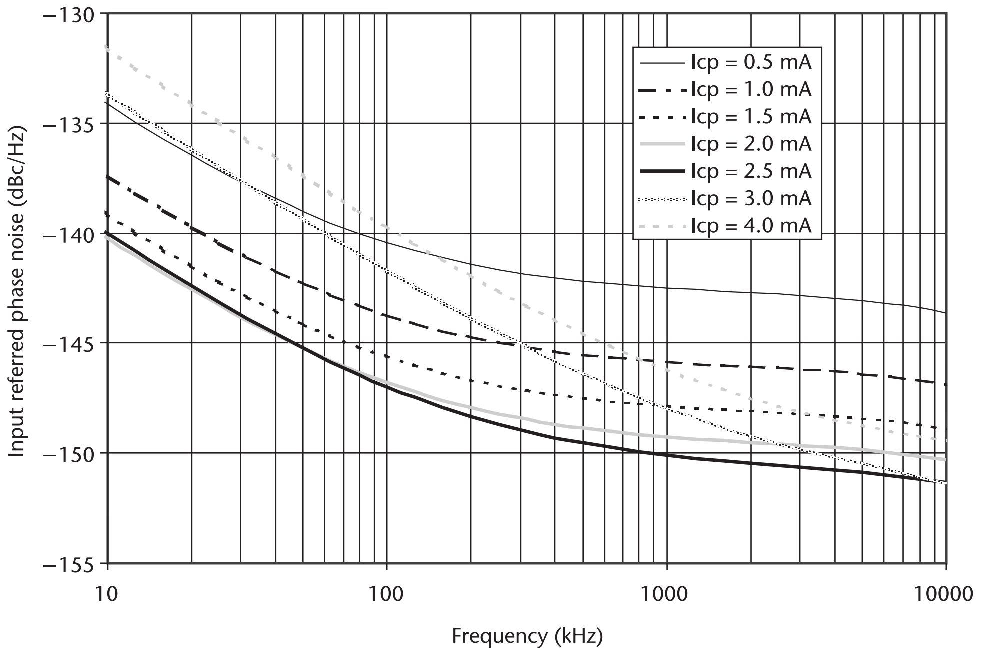


Figure 7.17 Input-referred phase noise for the charge pump example.

7.2.10 A Differential Charge Pump

From many perspectives, it is desirable to use differential circuits (especially on chip). In the past, it was difficult to build an LC VCO that had a differential control line, so the output of most charge pumps was required to be single ended. However, with modern processes, it is becoming possible to have LC VCOs that are controlled by differential signals. Techniques to do this will be discussed in Chapter 8. In applications that use ring oscillators, differential control lines are more common. For differential signals, common-mode and power-supply noise are rejected. However, to drive a differential VCO, a differential charge pump is needed. A basic circuit that can serve as a differential charge pump is shown in Figure 7.18 [7]. In this circuit, it is assumed that the current sources M_5, M_6, and M_8, M_9 are matched. When UP and DN are both low, the switches M_1 and M_4 are closed, and the current sourced from M_9 is all sunk by M_5; likewise, all current sourced by M_8 is sunk by M_6, and no current passes into or out of the loop filter (not shown, but connected between the output nodes). Now suppose that UP becomes active. In this case, M_1 is open, and M_2 is closed. Now the current from both M_9 and M_8 passes down the right-hand side of the circuit. Since M_6 can only sink half this current, the other half must pass through the loop filter and be sunk by M_5. In this case, a current is passed differentially through the loop filter, and a differential voltage is developed across it.

A similar, but opposite, situation occurs when a down pulse is present. Note that, even though this circuit resembles a CML circuit, it actually requires rail-to-rail signals in order to operate properly. Thus, if the PFD is made from CML, a rail-to-rail converter must be used to ensure that the switches operate for all output

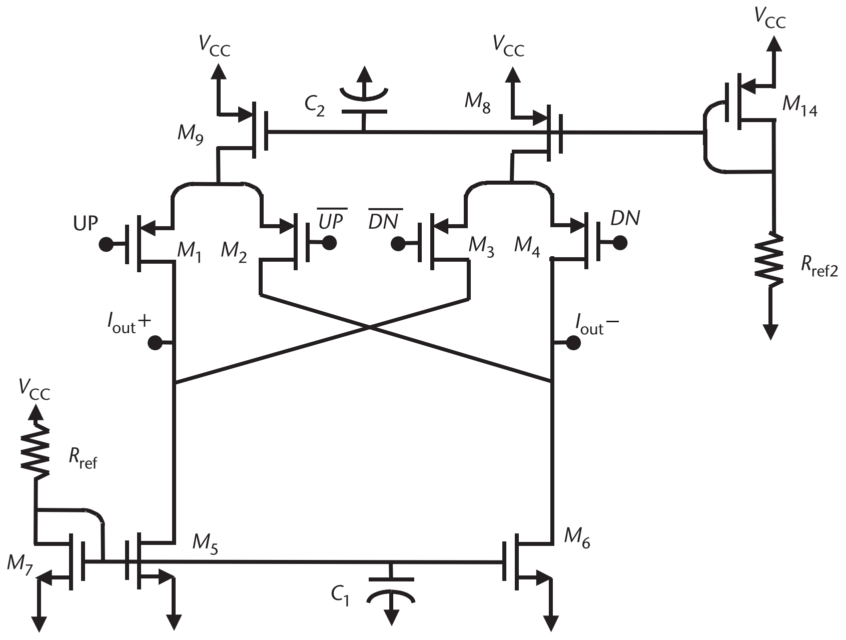


Figure 7.18 A basic differential charge pump circuit.

common-mode levels. Another disadvantage to this circuit is that it requires a static current flowing at all times to define the common-mode voltage on the output nodes. Thus, it is questionable as to whether this design can be as quiet as its single-ended counterparts. These circuits require a differential loop filter as well. A PLL with a basic differential loop filter is shown in Figure 7.19. It should be noted that by making C_2 differential, the loop filter nodes have high common-mode impedance, and since the loop only responds to differential feedback, there can be a lot of common-mode noise. Thus, it is better to break up the loop filter secondary capacitor and tie it explicitly to ground as shown.

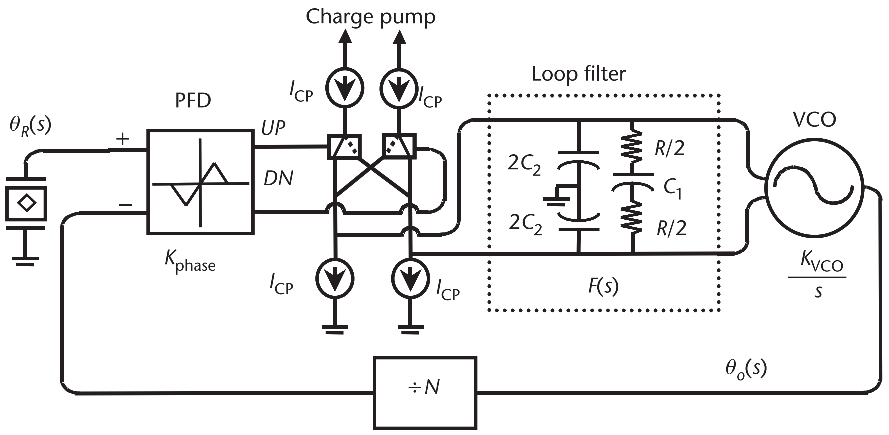


Figure 7.19 A synthesizer with a differential charge pump and loop filter.

7.2.11 Common-Mode Feedback for a Differential Charge Pump

During the previous discussion of differential charge pumps, one important issue has not been discussed. If the loop is to respond to a differential control signal, then what sets the common-mode voltage at the output of the charge pump? Obviously, the synthesizer cannot do this, so some additional circuitry is required. One such loop that could be used with a differential charge pump is shown in Figure 7.20. This circuit has two very large resistors labeled R_{CM} that sense the average voltage between the two outputs. These resistors must be very large so that they minimize the discharging of the loop filter. The voltage at the center of these two resistors is then compared to a reference voltage V_{ref}, the desired common-mode voltage. If the common-mode voltage is higher than V_{ref}, current is steered away from M_{14} by the differential pair M_{11} and M_{12}. If it is lower, more current is steered towards M_{14}. This current is then mirrored back to the current sources M_8 and M_9 to supply the current sources for the charge pump. This also automatically makes the up and down currents equal. Care must be taken to assure that this loop is stable. In Figure 7.20, stability is ensured by adjusting the value of capacitor C_2 to provide a dominant pole in the loop.

7.2.12 Another Differential Charge Pump

The one major drawback to the previous circuit is that it always draws current; therefore, its current sources are always on and, thus, producing noise. Another design that does not have common-mode current is shown in Figure 7.21. This differential charge pump is also CML-compatible. Now that there is no dc current flow, the need for a common-mode feedback loop (CMFB) loop is removed, but

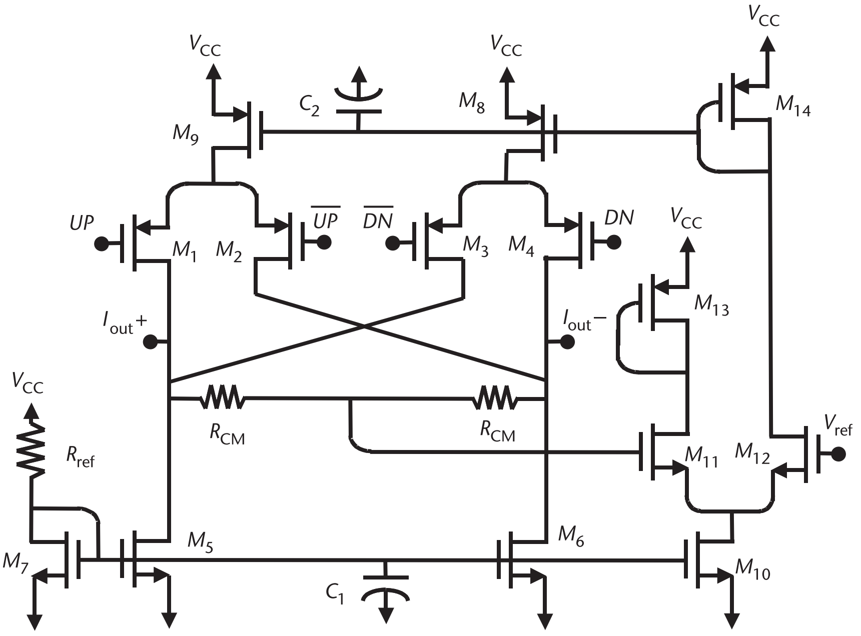


Figure 7.20 Differential charge pump with common-mode feedback circuitry added.

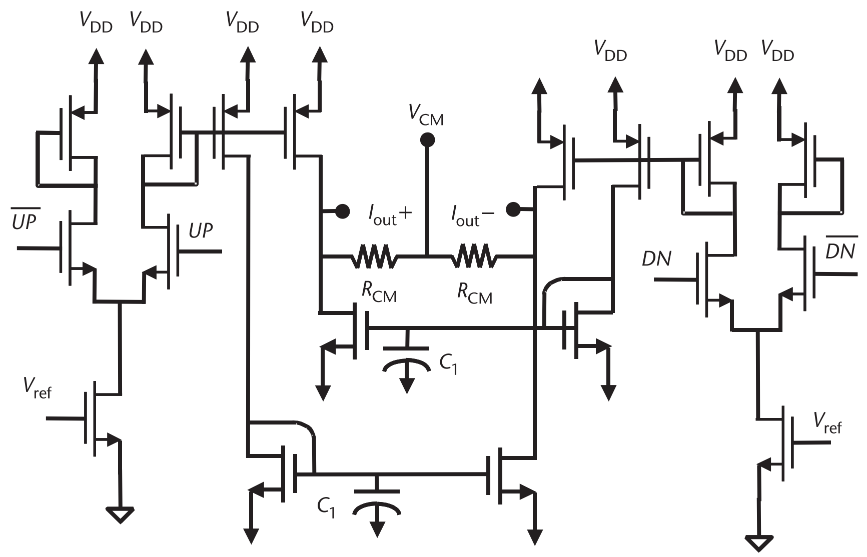


Figure 7.21 Differential charge pump with no dc current flow.

the common-mode voltage in this circuit still needs to be set. A common-mode voltage can be applied to the output through resistors R_{CM}. Again, these resistors must be large to minimize leakage current from the loop filter, thereby reducing reference spurs.

7.2.13 Programmable Bias Schemes

Often it is desirable to be able to adjust the current flowing in the charge pump either because the designer would like the flexibility of being able to adjust the loop bandwidth or because the exact charge pump current for best phase noise performance is not known with certainty before fabrication. The charge pump current can easily be made scalable with a simple circuit such as that shown in Figure 7.22. This circuit takes the reference current produced by the bandgap and then scales it up. Placing switches in series with the current mirrors allows the current to be programmed in binary steps, such that the output current I_{ref} is given by

$$I_{ref} = (8b_3 + 4b_2 + 2b_1 + b_0)I_{bias} \tag{7.27}$$

7.3 Loop Filters

The simplest second-order passive loop filter used with a charge pump has already been discussed in Chapter 3; however, there are also other loop filter implementations. In particular, higher-order loop filters are desirable if sharp roll-off of the loop transfer function is needed to reduce noise and spurs. This is especially important in fractional-N frequency synthesizers to suppress out-of-band fractional spurs. Higher-order loop filters can be implemented using pure passive components, such

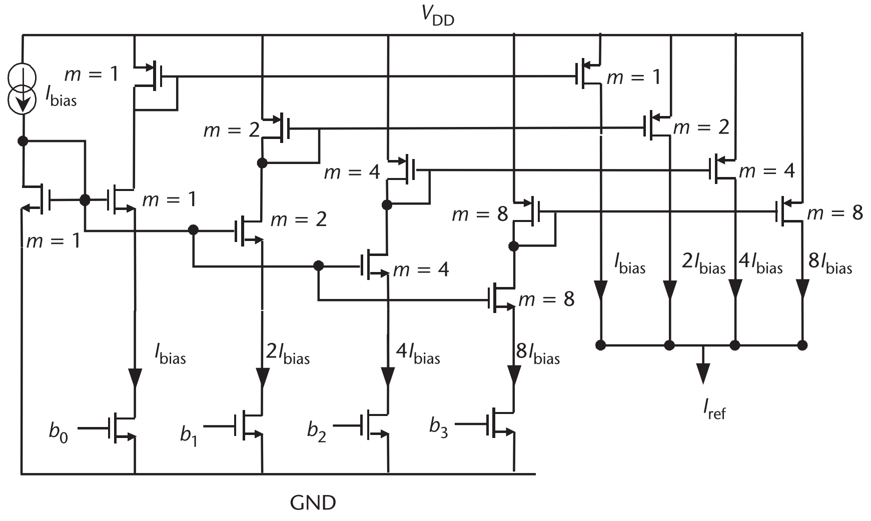


Figure 7.22 Charge pump bias circuitry with eight current settings programmable from I_{bias} to 15 I_{bias}.

as capacitors, resistors, and inductors. Passive components can also be used in combination with an operational amplifier to form an active loop filter. However, for stability, it is important that extra poles are well outside the loop bandwidth. As well, it should be noted that active components introduce additional noise that contributes to the in-band phase noise. Therefore, passive loop filters are normally employed to achieve low in-band noise. However, if the VCO requires a higher tuning voltage than the charge pump can provide, an active loop filter is necessary with the op-amp as a voltage buffer.

7.3.1 Passive Loop Filters

The simplest passive loop filter has already been discussed in Chapter 3 and is shown in Figure 7.1. The transfer function of the second-order passive loop filter is given by

$$F(s) = \frac{(1 + sC_1R)}{s(C_1 + C_2)(1 + sC_sR)} \tag{7.28}$$

where

$$C_s = \frac{C_1C_2}{C_1 + C_2} \tag{7.29}$$

Note that the filter transfer function is a transimpedance function, which transforms the charge pump output current into the VCO tuning voltage.

To achieve a higher-order transfer function, additional poles can be added using additional passive components. For instance, by adding a series resistor R_3, followed by a shunt capacitor C_3, the simple second-order filter can be modified to create a third-order passive filter, as shown in Figure 7.23. Note that the second-order filter is the highest-order passive RC filter that can be built without series resistors between the charge pump and the VCO tune line. For third- or higher-order passive RC filters, series resistors have to be added to introduce additional poles. The filter impedance of the third-order passive loop filter can be expressed as

$$F(s) = \frac{(1 + sT_1)}{sC_t(1 + sT_2)(1 + sT_3)} \tag{7.30}$$

where the total capacitance $C_t = C_1 + C_2 + C_3$, and the time constants are given by

$$\begin{cases} T_1 = R_1 \cdot C_1 \\ T_2 = R_1 \cdot C_1 \cdot C_2/C_t \\ T_3 \approx R_3 \cdot C_3 \end{cases} \tag{7.31}$$

It should be pointed out that the above approximation for calculating time constants is valid as long as

$$\begin{cases} C_i \ll C_1,\ i = 2,\ 3 \\ \dfrac{C_2}{C_3} + \dfrac{T_3}{T_1} \gg 1 \end{cases} \tag{7.32}$$

Detailed analysis shows that $T_2 + T_3 < T_1$ is required for stability; thus, the above approximation holds provided $C_3 < C_2/5$ [8].

Generalizing the third-order passive filter, Figure 7.24 illustrates a generic *k*th-order passive LPF for charge pump PLLs. For instance, the filter impedance for a fourth-order passive loop filter can be expressed as

$$F(s) = \frac{(1 + sT_1)}{sC_t(1 + sT_2)(1 + sT_3)(1 + sT_4)} \tag{7.33}$$

where the total capacitance $C_t = C_1 + C_2 + C_3 + C_4$, and the time constants are given by

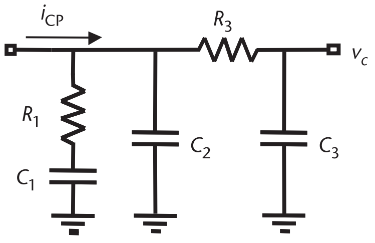


Figure 7.23 A third-order passive loop filter compatible with charge pump PLLs.

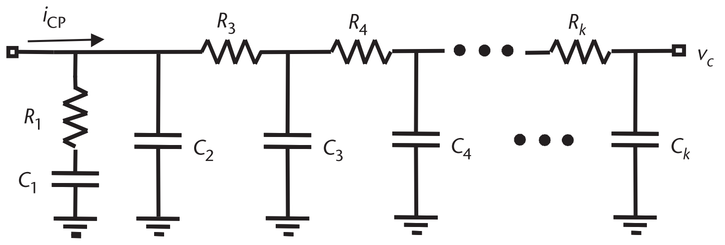


Figure 7.24 A generic *k*th-order passive LPF for charge pump PLLs.

$$\begin{cases} T_1 = R_1 \cdot C_1 \\ T_2 = R_1 \cdot C_1 \cdot C_2/C_t \\ T_3 \approx R_3 \cdot C_3 \\ T_4 \approx R_4 \cdot C_4 \end{cases} \tag{7.34}$$

In general, the filter impedance for a *k*th order passive loop filter shown in Figure 7.24 can be expressed as [8]

$$F(s) = \frac{(1 + sT_1)}{sC_t(1 + sT_2) \cdot \prod_{i=3}^{k} (1 + sT_i)} \tag{7.35}$$

where the total capacitance

$$C_t = \sum_{i=1}^{k} C_i$$

and the time constants are given by

$$\begin{cases} T_1 = R_1 \cdot C_1 \\ T_2 = R_1 \cdot C_1 \cdot C_2/C_t \\ T_i \approx R_i \cdot C_i \quad i = 3, 4, \ldots, k \end{cases} \tag{7.36}$$

It should be pointed out that the above approximation for calculating time constants is valid as long as

$$\begin{cases} C_i \ll C_1 \\ \dfrac{C_i}{C_{i+1}} + \dfrac{R_{i+1}}{R_i} \gg 1 \quad i = 3, 4, \ldots, k \end{cases} \tag{7.37}$$

The condition $T_i \geq 2T_{i+1}$ will satisfy the above constraints. A comparison of the open-loop gain and phase for second-, third-, and fourth-order filters is shown in Figure 7.25.

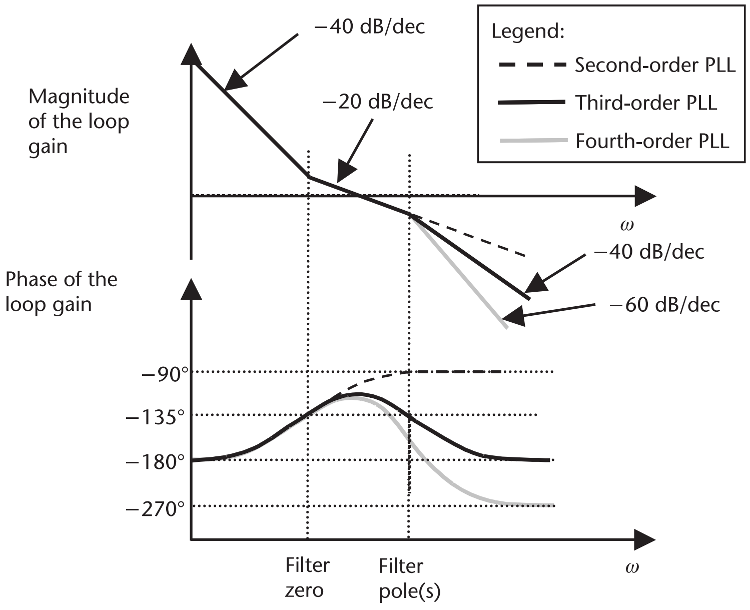


Figure 7.25 Comparison of open-loop gain and phase in a second-, third-, and fourth-order PLL.

7.3.2 Active Loop Filters

As shown in Figure 7.24, series resistors are required for third-order or higher passive RC loop filters. If series resistors are used in the loop filter, the VCO tuning voltage is lower than the charge pump output voltage, which is not desired. In applications like cable TV tuners, broadband tuning sometimes requires the VCO tuning voltage to be higher than that which the charge pump can provide. Under those circumstances, an op-amp can be added in the loop filter topology to transform the charge pump output voltage range to the desired VCO tuning range. In order to attenuate the added op-amp noise, it is recommended to use an active filter of third or higher order, even if it is not required for spur suppression. A simple third-order active loop filter is shown in Figure 7.26 [9, 10]. The presence of the op-amp has the added advantage that the output voltage on the charge pump is centered at $V_{DD}/2$. Thus, the charge pump no longer has to operate near the

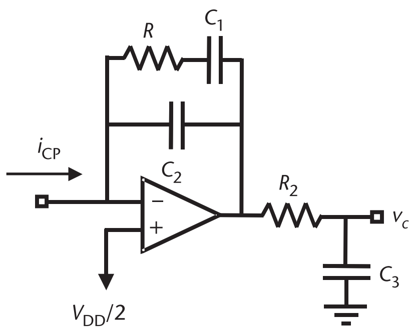


Figure 7.26 A third-order active loop filter compatible with charge pump PLLs.

rails, where the source or sink transistors are forced to operate in saturation for a BJT charge pump or in the triode region for a MOS charge pump.

The feedback path of the op-amp provides second-order lowpass characteristics, and an additional pole is added at the op-amp output. The filter transfer function is given by

$$F(s) = \frac{(1 + sC_1R_1)}{s(C_1 + C_2)(1 + sC_sR_1)(1 + sC_3R_3)} \tag{7.38}$$

where C_s is defined in (7.29). Generally, the third pole of the system is added to further reduce the out-of-band spurs; thus, it is placed at a higher frequency than the corner frequency of the loop. A good starting point for the relationship between the poles and zeros for this filter can be arranged as

$$C_1R_1 = 10C_sR_1 = 10C_3R_3 \tag{7.39}$$

Thus, C_2 is again chosen to be one-tenth the value of C_1. C_3 and R_3 are chosen so that this pole is at the same frequency as the higher-frequency pole in the simpler filter. In this manner, the loop will settle almost as quickly as a second-order loop (which will have the fastest settling) but still gain the advantage of additional filtering.

Adding an additional pole to the above third-order active filter, we can create a fourth-order active filter as illustrated in Figure 7.27. The impedance for the fourth-order active filter is given by

$$F(s) = \frac{(1 + sC_1R_1)}{s(C_1 + C_2)(1 + sC_sR_1)(1 + sT_3)(1 + sT_4)} \tag{7.40}$$

where the two high-frequency poles are located at

$$\begin{cases} T_3 = \frac{1}{2}\left(C_3R_3 + C_4R_3 + C_4R_4 + \sqrt{(C_3R_3 + C_4R_3 + C_4R_4)^2 - 4C_3C_4R_3R_4}\right) \\ T_4 = \frac{1}{2}\left(C_3R_3 + C_4R_3 + C_4R_4 - \sqrt{(C_3R_3 + C_4R_3 + C_4R_4)^2 - 4C_3C_4R_3R_4}\right) \end{cases} \tag{7.41}$$

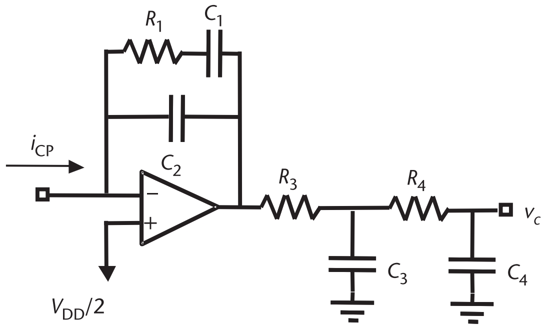


Figure 7.27 A fourth-order active loop filter compatible with charge pump PLLs.

An active loop filter can also be directly connected to the outputs of a different PFD as illustrated in Figure 7.28, where the charge pump is omitted. The differential voltage signals at the phase detector outputs are directly sensed by the op-amp. In this case, the op-amp replaces the charge pump function and provides a single-end voltage output that is proportional to its input voltage difference. This topology is very useful if the nonlinearity and noise of the charge pump are the dominant factors of the in-band phase noise. The impedance of the second-order active filter is given by

$$F(s) = \frac{(1 + sC_2R_2)}{sC_2R_1(1 + sC_3R_3)} \tag{7.42}$$

7.3.3 LC Loop Filters

In the previous sections, we were focused on RC loop filter designs. These types of filters are limited to transfer characteristics with real poles whose roll-off slope is 20 dB/dec per order. Since the typical cutoff frequency of the PLL loop filter is in the range of a few kilohertz to a few megahertz, the filter component values are normally large such that on-chip filters normally occupy a large amount of silicon area. Therefore, the loop filters are often off chip, which not only saves die area but also provides flexibility of filter design. If the loop filter is off chip and area is no longer a concern, LC filters can be employed to achieve various roll-off characteristics. Also, LC filters have lower noise compared to RC filters. A typical LPF configuration using an LC ladder is shown in Figure 7.29, where the order of the filter is determined by the number of elements placed in the ladder. The order increases by two when a pair of LC elements is added to the ladder. For charge pump-based PLLs, a transimpedance stage is still required to convert the charge pump current to a voltage. For this purpose, an op-amp can be added in front of the filter to form an active LC filter, as shown in Figure 7.29. The op-amp provides a low impedance source for the LC ladder filter. Since the load of the LC ladder is the VCO tuning input, which is a high impedance, a resistor is added at the output to define the response of the circuit. Note that the input op-amp provides a dominant second-order pole to the system. The poles of the LC

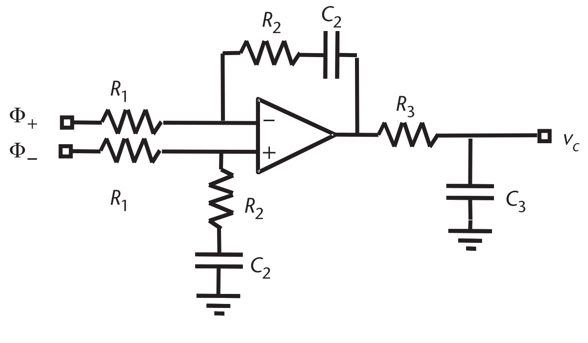


Figure 7.28 A second-order active loop filter compatible with a differential PFD.

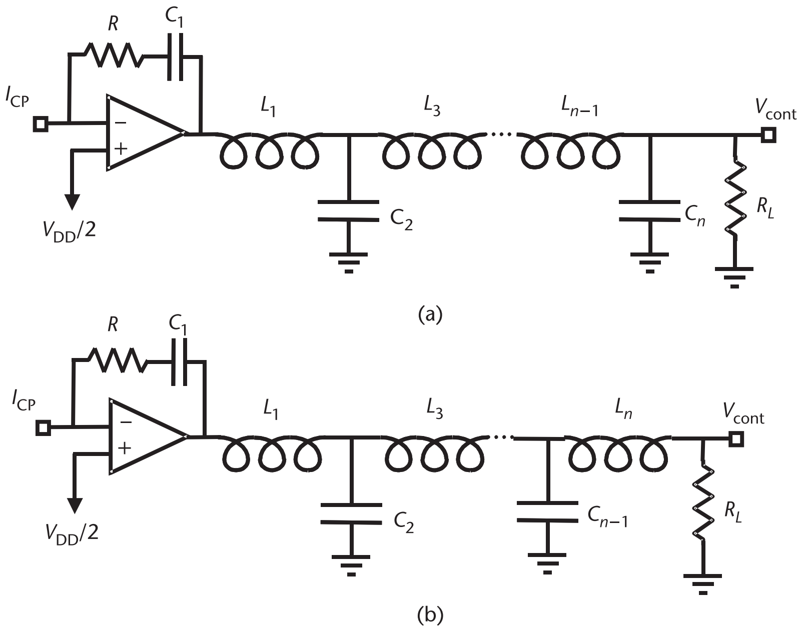


Figure 7.29 High-order loop filters implemented with passive LC ladders: (a) filter order *n* is an even number, and (b) *n* is an odd number.

ladder filter are then added at higher frequencies to provide out-of-band spur reduction. As always, a second-order transient response is desired to ensure fast settling and to avoid stability problems.

In order to size the filter components, the desired filter response has to be determined. The most commonly used filter transfer functions are Butterworth and Chebyshev. The filter transfer functions differ in their passband ripple and their roll-off slope. Butterworth filters of order n have maximally flat responses in the passband, with roll-off slope of $20n$ dB/dec. The attenuation of a Butterworth filter is given by

$$A_{\text{Butterworth}}\,(\text{dB}) = 10\log\left[1+\left(\frac{\omega}{\omega_c}\right)^{2n}\right] \tag{7.43}$$

where n is the filter order, and ω_c is the cutoff frequency. On the other hand, Chebyshev filters allow ripple in the passband; in return, Chebyshev filters achieve sharper roll-off in the stopband. A Chebyshev filter's attenuation is given by

$$A_{\text{Chebyshev}}\,(\text{dB}) = 10\log\left\{1 + r^2\cdot C_n^2\left[\cosh B\cdot\left(\frac{\omega}{\omega_c}\right)\right]\right\} \tag{7.44}$$

where C_n is the nth order Chebyshev polynomial given by

$$C_3 = 4\left(\frac{\omega}{\omega_c}\right)^3 - 3\frac{\omega}{\omega_c}$$

$$C_4 = 8\left(\frac{\omega}{\omega_c}\right)^4 - 8\left(\frac{\omega}{\omega_c}\right)^2 + 1$$

$$C_5 = 16\left(\frac{\omega}{\omega_c}\right)^5 - 20\left(\frac{\omega}{\omega_c}\right)^3 + 5\frac{\omega}{\omega_c} \tag{7.45}$$

$$C_6 = 32\left(\frac{\omega}{\omega_c}\right)^6 - 48\left(\frac{\omega}{\omega_c}\right)^4 + 18\left(\frac{\omega}{\omega_c}\right)^2 - 1$$

$$C_7 = 64\left(\frac{\omega}{\omega_c}\right)^7 - 112\left(\frac{\omega}{\omega_c}\right)^5 + 56\left(\frac{\omega}{\omega_c}\right)^3 - 7\frac{\omega}{\omega_c}$$

for third- to seventh-order filters. The passband ripple r is given by

$$r = \sqrt{10^{r(\text{dB})/10} - 1} \tag{7.46}$$

and B is given by

$$B = n^{-1}\cosh^{-1}(r^{-1}) \tag{7.47}$$

It can be shown that a Chebyshev filter with more ripple in the passband will have a steeper roll-off in the stopband. Figure 7.30 shows a comparison of the amplitude response of these filters.

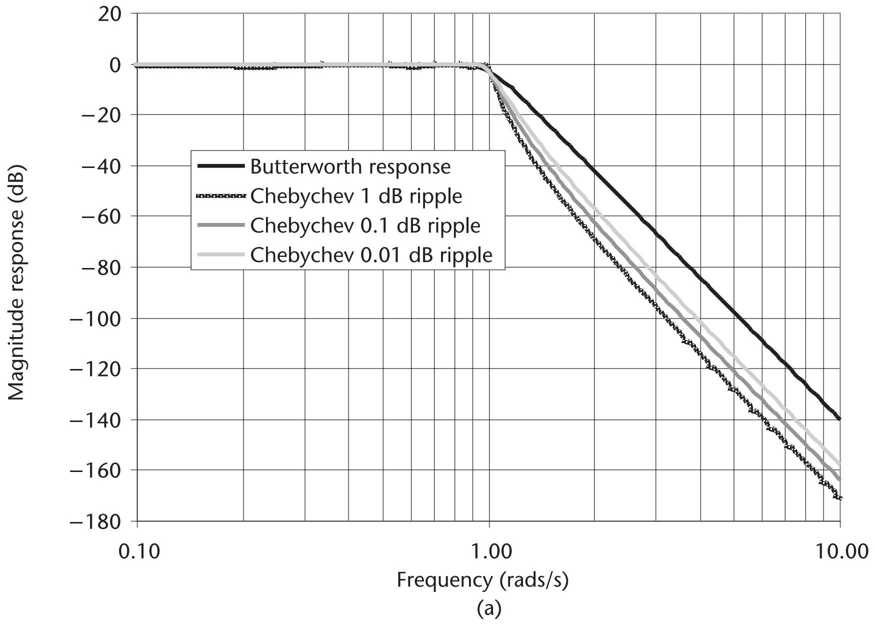
Table 7.1 gives the filter component values to implement Butterworth and Chebyshev LPFs using the LC ladder shown in Figure 7.29. Note that the component values listed in the table are normalized to a 1-rad/s cutoff frequency. The values can be scaled to any desired cutoff frequency by using the following formulas:

$$C_{\text{final}} = \frac{C_{\text{norm}}}{2\pi f_c R_L} \tag{7.48}$$

and

$$L_{\text{final}} = \frac{R_L L_{\text{norm}}}{2\pi f_c} \tag{7.49}$$

where f_c is the desired filter cutoff frequency, and C_{norm} and L_{norm} are the respective capacitor or inductor values given in Table 7.1. This table has been prepared with the assumption that the load impedance R_L is much larger than the source impedance (i.e., the output impedance of the op-amp for the active filter shown in Figure 7.29). For filter component values with other load-to-source impedance ratios, the reader can refer to [11].



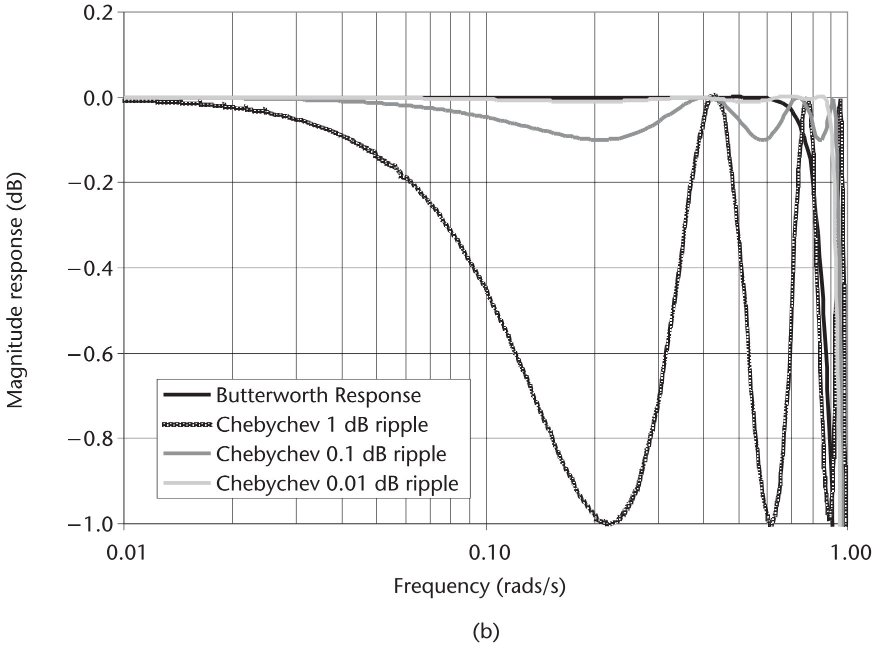


Figure 7.30 Amplitude response for seventh-order Butterworth and Chebyshev filters normalized to 1 rad/s in (a) the stopband, and (b) the passband.

Example 7.3: Loop Filter Design

Modify the loop filter in Example 7.1 to achieve greater reference spur rejection. Characterize the spur rejection assuming that a charge pump mismatch of 30% exists in the design. You may use inductors, but attempt to keep the size under 1 mH.

Table 7.1 Normalized Component Values for Butterworth and Chebyshev LC Lowpass Filters

Filter Type	*Filter Order*	L_1	C_2	L_3	C_4	L_5	C_6	L_7
Butterworth	3	1.500	1.333	0.500				
	4	1.531	1.577	1.082	0.383			
	5	1.545	1.694	1.382	0.894	0.309		
	6	1.553	1.759	1.553	1.202	0.758	0.259	
	7	1.558	1.799	1.659	1.397	1.055	0.656	0.223
Chebyshev with 0.01-dB ripple	3	1.501	1.433	0.591				
	4	1.529	1.694	1.312	0.523			
	5	1.547	1.795	1.645	1.237	0.488		
	6	1.551	1.847	1.790	1.598	1.190	0.469	
	7	1.559	1.867	1.866	1.765	1.563	1.161	0.456
Chebyshev with 0.1-dB ripple	3	1.513	1.510	0.716				
	4	1.511	1.768	1.455	0.673			
	5	1.561	1.807	1.766	1.417	0.651		
	6	1.534	1.884	1.831	1.749	1.394	0.638	
	7	1.575	1.858	1.921	1.827	1.734	1.379	0.631
Chebyshev with 1-dB ripple	3	1.652	1.460	1.108				
	4	1.35	2.01	1.488	1.106			
	5	1.721	1.645	2.061	1.493	1.103		
	6	1.378	2.097	1.69	2.074	1.494	1.102	
	7	1.741	1.677	2.155	1.703	2.079	1.494	1.102

Solution: In order to get higher spur rejection, we need to use a higher-order loop filter. In order to maximize spur rejection, a seventh-order passive LC ladder filter, as illustrated in Figure 7.29, will be chosen for implementation. Now we must choose a value for the load resistor in this design. Looking at (7.49), we notice that, as the load resistance is increased, the inductors in the design get larger. Thus, to keep the component values small, we need to reduce the load resistance. Since there must be a dc path from the output of the op-amp to the input of the VCO, the op-amp must be able to provide dc current to the load resistor. If we choose R_L as 1 kΩ and bias the other end of this resistor at $V_{DD}/2$, then the op-amp may have to provide as much as 1.5 mA of current. Note that a decoupling capacitor could fix this at the expense of one extra component.

Note that, in real life, the filter output will drive the varactors in the VCO. Typical values for such varactors will be in the 1- to 5-pF range. Past the corner frequency of the load resistor and the varactor capacitance, the ladder is no longer loaded properly with R_L and will no longer perform as expected. Since the ladder filter must filter reference spurs, in this case at 40 MHz, such corner should be above 40 MHz. At 5 pF, this means that the resistance should be less than about 800Ω, or, with a load resistance of 1,000Ω, the capacitor should be less than about 4 pF.

Now we want the LC ladder to replace the function of the capacitor C_2, so this part of the filter is chosen to have a cutoff frequency $f_c = 1.5$ MHz (about 10 times higher than the loop bandwidth). Thus, we choose in this design $C_1 = 5.66$ nF and $R = 530\Omega$ just as in Example 7.1. Using (7.48) and (7.49) and making use of Table 7.1, we can find $L_1 = 185\ \mu$H, $C_2 = 178$ pF, $L_3 = 229\ \mu$H, $C_4 = 181$ pF,

L_5 = 221 μH, C_6 = 159 pF, and L_7 = 117 μH, assuming that we choose a Chebyshev filter with 1-dB in-band ripple.

Now the filter response of the two designs can be compared. The relative response is shown in Figure 7.31. From this graph, it can be seen that, in band, the response of the two filters is almost identical, but out of band (beyond about 1.6 MHz), the LC ladder filter has dramatically more attenuation than the simple filter. The fact that the in-band response is similar means that both filters should have the same transient response to a frequency step at the output. Figure 7.32 shows that the PLL using both types of loop filter leads to a very similar settling behavior for a step in frequency of 20 MHz at the output. Note that, with the LC

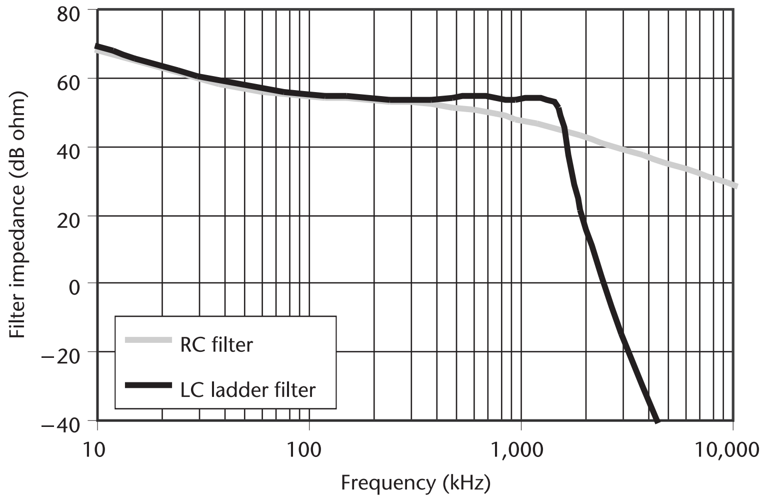


Figure 7.31 Comparison of a standard loop filter with a high-order filter using an LC ladder and op-amp design.

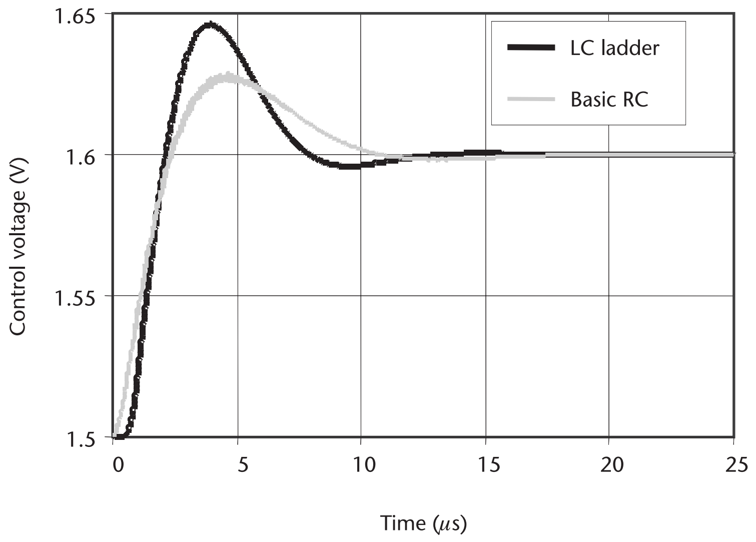


Figure 7.32 Comparison of transient response of a PLL using standard and high-order loop filters.

ladder filter, there is slightly more ringing, and settling takes slightly longer than expected. This difference in transient behavior can be reduced by increasing the corner frequency of the LC filter but at the expense of reduced suppression of reference feedthrough. Finally, the reference spur rejection can be compared for both designs with a 30% charge pump current mismatch. This means that the UP current source is now set to 816.4 μA, and the PFD delay remains at 5 ns. Comparison of the designs is shown in Figure 7.33. From this graph, it can be seen that the RC filter design has reference spurs that are 59.6 dB below the carrier, while the LC ladder spurs are below the noise floor of the simulation. Thus, they are lower than 103 dB below the carrier, and the LC ladder design provides at least an additional 43 dB of reference spur rejection.

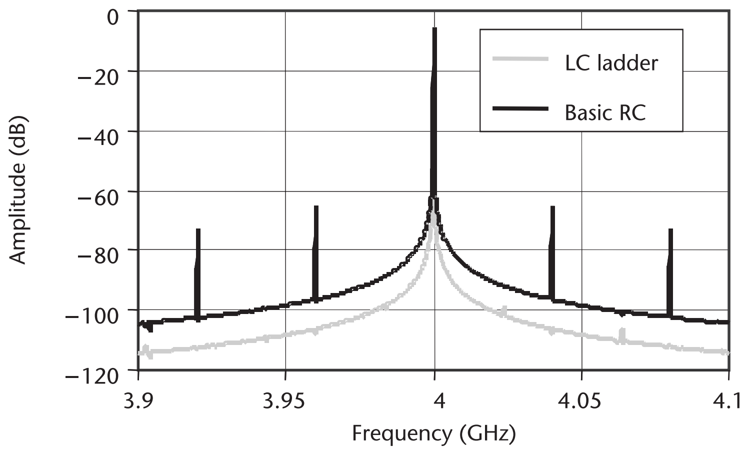


Figure 7.33 Comparison of reference spur rejection of a PLL using standard and high-order loop filter.

References

[1] Gray, P. R., et al., *Analysis and Design of Analog Integrated Circuits*, 4th ed., New York: John Wiley & Sons, 2001.

[2] Johns, D. A., and K. Martin, *Analog Integrated Circuit Design*, New York: John Wiley & Sons, 1997.

[3] Hung, C., and K. K. O, "A Fully Integrated 1.5V 5.5 GHz CMOS Phase-Locked Loop," *IEEE J. Solid-State Circuits*, Vol. 37, No. 4, April 2002, pp. 521–525.

[4] Terrovitis, M., et al., "A 3.2 to 4 GHz 0.25 μm CMOS Frequency Synthesizer for IEEE 802.11a/b/g WLAN," *ISSCC Dig. Tech Papers*, San Francisco, CA, February 2004, pp. 98–99.

[5] Rogers, J. W. M., et al., "A Fully Integrated Multi-Band $\Sigma\Delta$ Fractional-*N* Frequency Synthesizer for a MIMO WLAN Transceiver RFIC," *IEEE J. Solid-State Circuits*, Vol. 40, No. 3, March 2005, pp. 678–689.

[6] Lee, J., and B. Kim, "A Low-Noise Fast-Lock Phase-Locked Loop with Adaptive Bandwidth Control," *IEEE J. Solid-State Circuits*, Vol. 35, No. 8, August 2005, pp. 1137–1145.

[7] Razavi, B., *Monolithic Phase-Locked Loops and Clock Recovery Circuits*, New York: Wiley-IEEE Press, 1996.

[8] Banerjee, D., *PLL Performance, Simulation and Design*, 3rd ed., San Jose, CA: Dean Banerjee Publications, 2003.

[9] Mijuskovic, D., et al., "Cell-Based Fully Integrated CMOS Synthesizers," *IEEE J. Solid-State Circuits*, Vol. 29, March 1994, pp. 271–279.

[10] Craninckx, J., and M. S. J. Steyaert, "A Fully Integrated CMOS DCS-1800 Frequency Synthesizer," *IEEE J. Solid-State Circuits*, Vol. 33, December 1998, pp. 2054–2065.

[11] Bowick, C., *RF Circuit Design*, Burlington, MA: Newnes, 1982.

CHAPTER 8

Voltage-Controlled Oscillators

8.1 Introduction

An oscillator is a circuit that generates a periodic waveform. While oscillators have numerous applications, from serving as reference tone generators for receivers to clocks for digital circuits, the application of most importance for this text is their use as the central component in a frequency synthesizer. In this application, important design considerations include tunability, spectral purity (low phase noise), and, in the fully integrated context, low power. Where the lowest possible phase noise with low power dissipation is the driving consideration, designers may decide to use LC-based oscillators. However, in many situations, ring oscillators are used; since they can potentially provide wider tuning range, they are simpler to design in a standard digital process as, typically, no inductors or varactors are needed. Without inductors, their layout area is also significantly lower. In addition, it is possible to design ring oscillators in such a way that they automatically have quadrature outputs. This chapter considers crystal oscillators, in addition to LC and ring oscillators. Crystal oscillators are important since they are commonly used as reference signals for frequency synthesizers.

8.2 Specification of Oscillator Properties

Possibly the most important characteristic of an oscillator is its phase noise. An ideal oscillator has accurate periodicity with all signal power concentrated in one discrete oscillator frequency (possibly at multiples of the oscillator frequency). However, all real oscillators have less-than-perfect spectral purity, having phase noise, which is seen as undesired frequency components around the desired frequency, as shown in Figure 8.1. It is desirable to minimize these extra frequency components as much as possible. Chapter 3 discusses the effect of oscillator phase noise on synthesizer performance. Other considerations are long-term stability and sufficient output voltage amplitude for the intended application.

8.3 LC-Based VCOs

The LC resonator at the core of LC oscillators determines the frequency of oscillation and often forms part of the feedback mechanism used to obtain sustained oscillations. The frequency of oscillation will be given approximately by

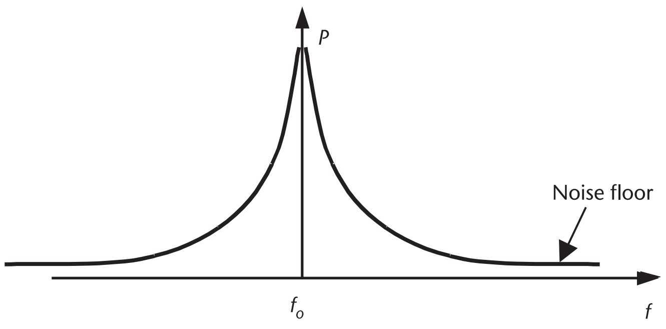


Figure 8.1 Spectrum of a typical oscillator.

$$\omega_{\mathrm{osc}} = \sqrt{\frac{1}{LC}} \tag{8.1}$$

Any oscillation, once started, will tend to decay due to the losses in the resonator. In order to maintain sustained oscillations, it is necessary to provide feedback to restore energy. Two popular ways to achieve this are by the Colpitts oscillator or the negative G_m oscillator. These can be shown with bipolar or MOS transistors; here we show the MOS versions. Note that with MOS, one can choose NMOS or PMOS. NMOS will have higher g_m for the same transistor size and current; however, PMOS will have lower $1/f$ noise. Differential versions of these two oscillators are shown in Figure 8.2. Note that the trend is for CMOS oscillators to be nearly exclusively of the negative G_m variety because of their simplicity. However, Colpitts oscillators are not completely extinct, especially with bipolar transistors, so some knowledge of them can be valuable.

8.3.1 Inductors

Of all the passive structures used in RF circuits, high-quality inductors are traditionally among the most difficult to realize monolithically. In silicon, they suffer from the presence of lossy substrates and high-resistivity metal. However, over the past few years, much research has been done into efforts to improve fabrication methods for building inductors, as well as modeling, so that better geometries can be used in their fabrication [1–4].

Traditionally, due to limitations in modeling and simulation tools, inductors were made as square spirals. The wrapping of the metal lines allows the flux from each turn to add, thus increasing the inductance per unit length of the structure and providing a compact way of achieving useful values of inductance. Square inductors, however, have less-than-optimum performance due to the 90° bends present in the layout, which add to the resistance of the structures. A better structure is shown in Figure 8.3. Since this inductor is circular, it has less series resistance. This geometry is more symmetric than traditional inductors (its S-parameters look the same from either side). Thus, it can be used in differential circuits (like most VCOs) without needing two inductors to get good symmetry. Bias can be applied through the axis of symmetry of this structure if needed in a differential application

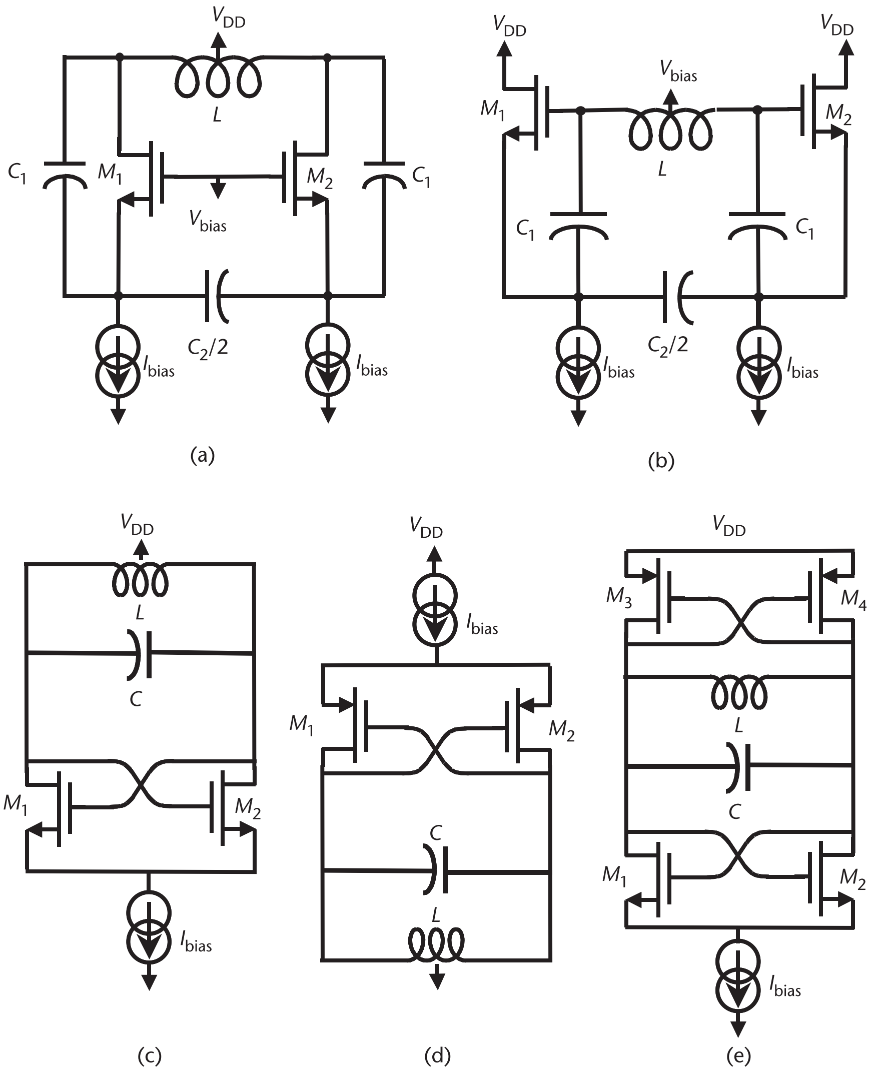


Figure 8.2 Basic differential oscillators: (a) Colpitts common gate, (b) Colpitts common drain, (c) $-G_m$ oscillator (NMOS only), (d) $-G_m$ oscillator (PMOS only), and (e) $-G_m$ oscillator (complementary).

(i.e., it is a virtual ground point). Note that most technologies do not offer truly circular structures but get almost the same performance with octagonal structures.

When describing on-chip inductors, it is useful to build an equivalent model for the structure. Figure 8.4 shows capacitance between lines, capacitance through the oxide, inductance of the traces, series resistance, and substrate effects. These effects are translated into the circuit model shown in Figure 8.5, which shows a number of nonideal components. R_s models the series resistance of the metal lines used to form the inductor. Note that the value of R_s will increase at higher frequencies due to the skin effect, which is an electromagnetic (EM) effect that causes current to crowd near the edges of the conductor. C_{oxide} models the capacitance from the lines to the substrate. This is essentially a parallel-plate capacitor formed

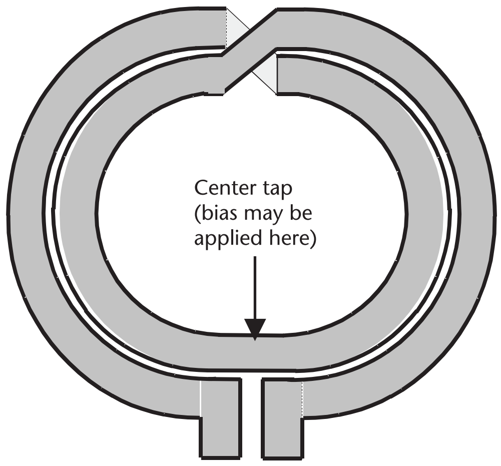


Figure 8.3 A circular differential inductor layout.

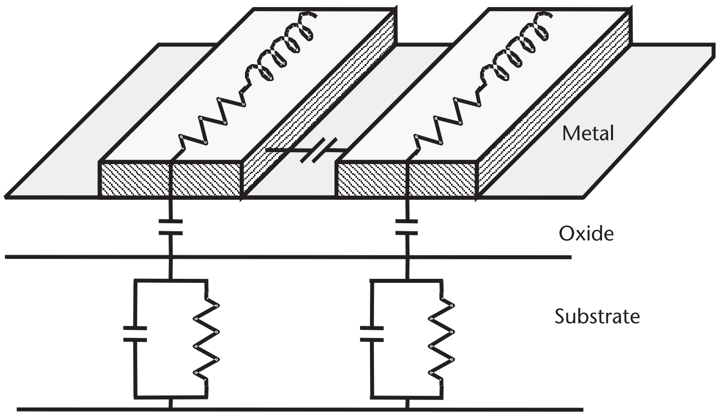


Figure 8.4 Illustration of elements used to build an inductor model.

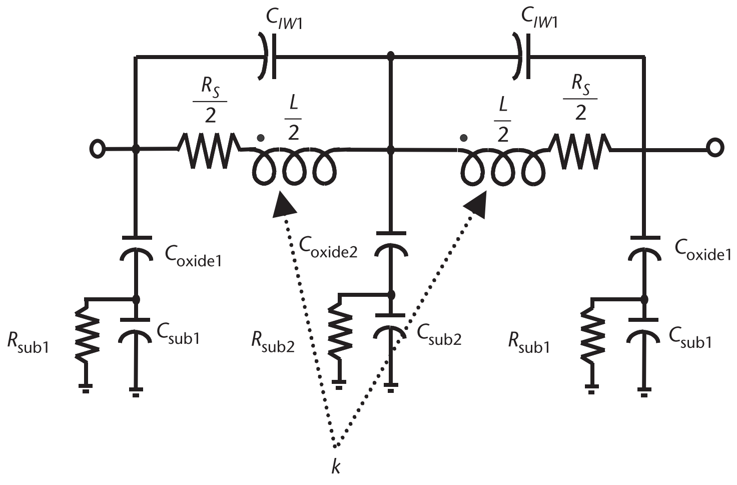


Figure 8.5 Basic model for a differential inductor.

between the inductor metal and the substrate. C_{sub} and R_{sub} model the losses due to magnetic effects, capacitance, and the conductance of the substrate. They are proportional to the area of the metal in the inductor, and their exact value depends on the properties of the substrate in question. C_{IW} models the interwinding capacitance between the traces. This is another parallel-plate capacitor formed by adjacent metal lines. The model is broken into two parts with a pin at the axis of symmetry, where a bias can be applied if desired. Note also that since the two halves of the spiral are interleaved, there is magnetic coupling between both halves of the device. This is modeled by the coupling coefficient k.

At low frequencies, the inductance of an integrated inductor is relatively constant. However, as the frequency increases, the impedance of the parasitic capacitance elements starts to become significant. In fact, at some frequency, the admittance of the parasitic elements will cancel that of the inductor, and the inductor will self-resonate. At this point, the reactive part of the admittance will be zero. The inductance is nearly constant at frequencies much lower than the self-resonance frequency; however, as the self-resonance frequency is approached, the inductance rises and then abruptly falls to zero. Beyond the self-resonant frequency, the parasitic capacitance will dominate, and the inductor will look capacitive. Thus, the inductor has a finite bandwidth over which it can be used. For reliable operation, it is necessary to stay well below the self-resonance frequency. Since parasitic capacitance increases in proportion to the size of the inductor, the self-resonant frequency decreases as the size of the inductor increases. Thus, the size of on-chip inductors that can be built is limited.

The quality factor Q of a passive circuit element can be defined as

$$Q = \frac{|\mathrm{Im}(Z_{\text{ind}})|}{|\mathrm{Re}(Z_{\text{ind}})|} = \frac{\omega L}{r_s} = \frac{|\mathrm{Im}(Y_{\text{ind}})|}{|\mathrm{Re}(Y_{\text{ind}})|} = \frac{r_p}{\omega L} \tag{8.2}$$

where Z_{ind} is the impedance of the inductor, Y_{ind} is the admittance of the inductor, L is the equivalent inductance at frequency ω, and r_s and r_p are the equivalent series and parallel resistance of the inductor at frequency ω. This is not necessarily the most fundamental definition of Q, but it is a good way to characterize the structure. A good way to think about this is that Q is a measure of the ratio of the desired quantity (inductive reactance) to the undesired quantity (resistance). Obviously, the higher the Q, the more ideal the device becomes.

The Q of an on-chip inductor is affected by many things. At low frequencies, the Q tends to increase with frequency because the losses are relatively constant (mostly due to metal resistance R_s), while the imaginary part of the impedance is increasing linearly with frequency. However, as the frequency increases, currents start to flow in the substrate through capacitive and, to a lesser degree, magnetic coupling. This loss of energy into the substrate causes an effective increase in the resistance. In addition, the skin effect starts to raise the resistance of the metal traces at higher frequencies. Thus, most integrated inductors have Qs that rise at low frequencies; they then have some peak beyond which the losses make the resistance rise faster than the imaginary part of the impedance, and the Q starts to fall off again. Thus, it is easy to see the need for proper optimization to ensure that the inductor has peak performance at the frequency of interest.

8.3.2 Varactors for Oscillator Frequency Control

Typically, one has the choice of a few different kinds of varactors [5]. The first kind, the pn varactor, is formed from a pn junction (often inside of a well), as shown in Figure 8.6(a). Typically, such varactors have a parasitic varactor to the substrate. Unlike the desired pn junction, which has a high Q, this parasitic junction has a low Q due to the lower doping of the substrate. This makes it desirable to remove it from the circuit. This can be done by placing the varactors in the circuit such that the side with the parasitic diodes is tied together at the axis of symmetry, as shown in Figure 8.6(b).

A varactor can also be formed from a regular MOS transistor where the gate is one terminal and the source and drain tied together form the other terminal. Such a structure is shown in Figure 8.7. As a positive voltage is put on the gate, holes in the p substrate are driven away from the surface, forming a depletion region whose depth depends on applied voltage. At a large-enough positive voltage, an inversion layer of electrons forms along the surface, and the capacitance is at its maximum value, that of the gate oxide.

If a negative voltage is applied, a layer of holes is accumulated next to the gate oxide. This might be expected to increase the capacitance. However, if the substrate is not connected to the source, there is no direct electrical connection between the

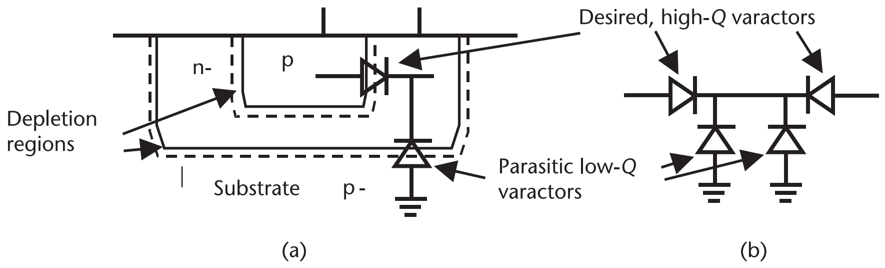


Figure 8.6 (a) Desired pn varactor and parasitic pn varactor and (b) connecting two pn varactors in such a way that parasitic varactors are at the common-mode point.

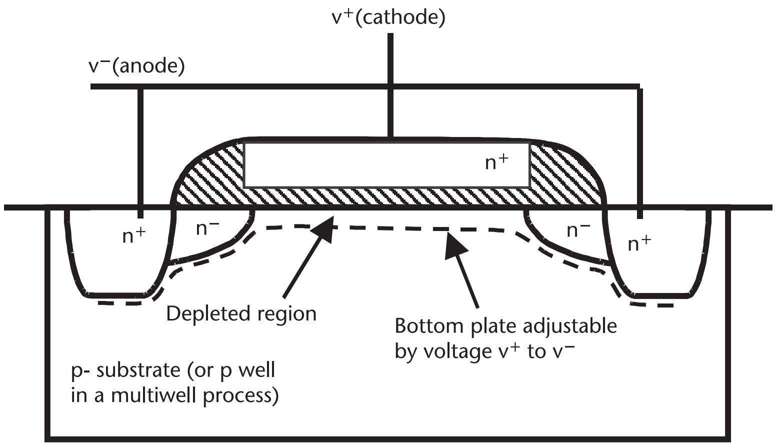

v^+(cathode)
v^-(anode)
n^+
n^+
n^-
n^-
n^+
Depleted region
Bottom plate adjustable by voltage v^+ to v^-
p- substrate (or p well in a multiwell process)

Figure 8.7 Cross section of a MOS varactor in the depletion region.

accumulation region and the source and drain regions. With a negative gate voltage, there is now a relative positive voltage on the source and drain; as a result, there is a substantial depletion region along the source and drain, as shown in Figure 8.8. This depletion capacitance is in series with gate capacitance; as a result, the capacitance does not increase back up to gate capacitance as might have been expected. In this region, the low doping in the substrate provides a lossy signal path; hence, the *Q* of such varactors can be quite low.

Another technique is to use the gate as one terminal and the substrate connection as the other. In such a case, source and drain are not required. Such a varactor is often called the accumulation MOS (AMOS) varactor since the capacitance is highest when the surface under the gate is in accumulation. Capacitance is lowest in the inversion region. As with all types of MOS varactors, the AMOS varactor takes advantage of the variation of depletion capacitance with applied voltage. Operation of the AMOS varactor, shown in Figure 8.9, will now be described.

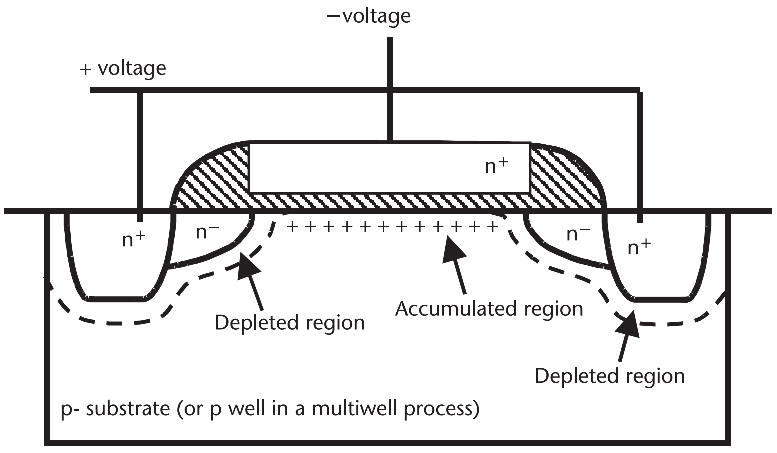


Figure 8.8 Cross section of a MOS varactor shown in the accumulation region.

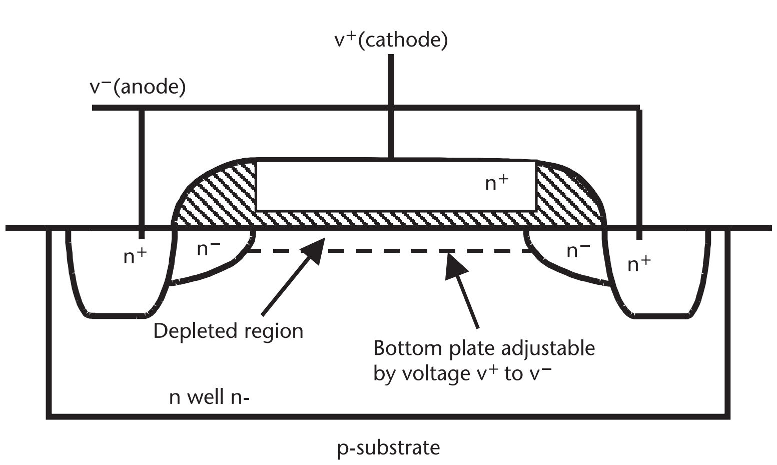


Figure 8.9 Cross section of typical AMOS varactor in the depletion region.

A typical varactor characteristic is shown in Figure 8.10 [6]. The different regions of operation are illustrated in Figure 8.11. When sufficient positive voltage is applied to the gate, a layer of negative charges (electrons) forms along the oxide surface, providing a second electrode to the capacitor and a maximum capacitance equal to the gate capacitance. As the voltage is decreased, this layer of charge disappears and a depletion region forms under the gate oxide. In this region, the depletion width, hence, the capacitance, is dependent on the gate voltage. For sufficiently large negative voltage on the gate, holes are attracted to the oxide surface, and the capacitor is said to be in inversion. In this region, the layer of holes prevents the depletion region from growing any further, and the total capacitance is the series capacitance between the gate oxide and the depletion capacitance. Note

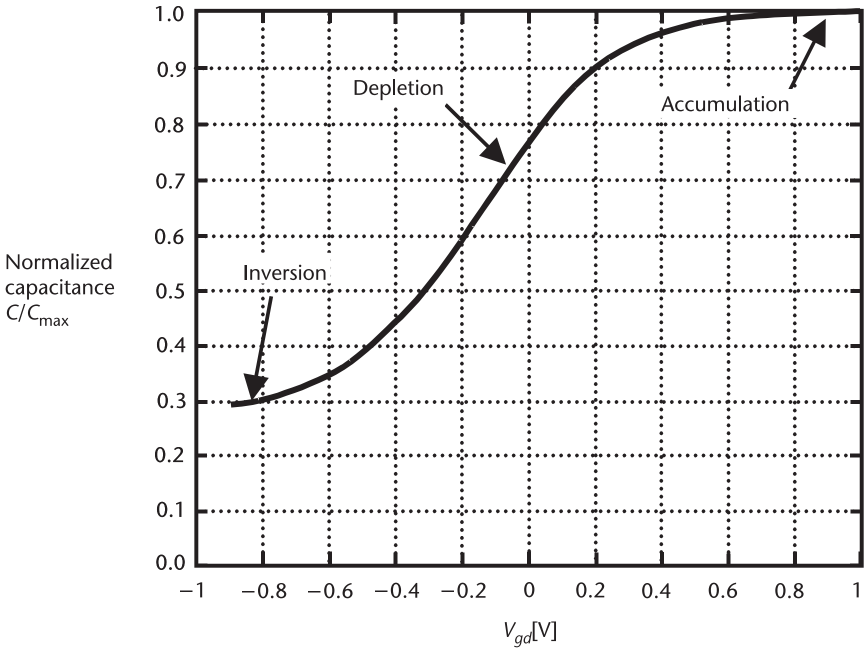


Figure 8.10 Example of a capacitance versus voltage curve of an AMOS varactor.

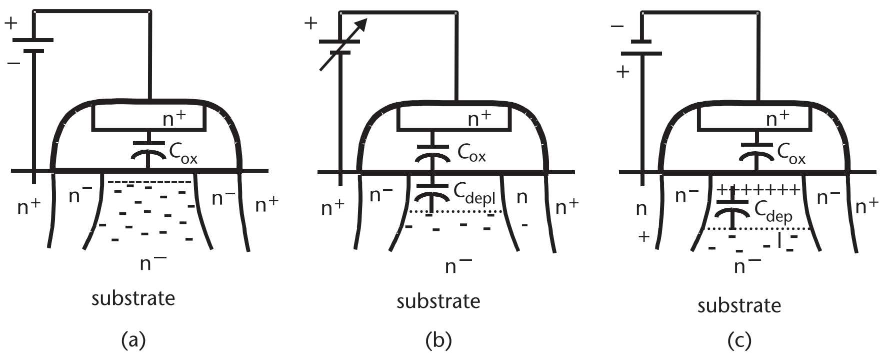


Figure 8.11 Regions of operation in an AMOS varactor: (a) accumulation; (b) depletion; and (c) inversion.

that while the holes are conductive, they do not connect to the n-type anode, and, hence, do not short out the depletion capacitance.

According to [7], using minimum channel length results in the highest-Q varactors since this minimizes the series resistance. However, with minimum length, the fixed capacitors, such as overlap capacitance, are more important relative to the variable capacitors, so this reduces the ratio of maximum to minimum capacitance. As for Q as a function of region of operation, the series resistance is lowest in the accumulation region, where there is a conductive layer close to the gate oxide. As the depletion region is entered, the resistance R will increase; however, in this region, the capacitance is decreasing, and capacitive reactance X_C is increasing. Since Q is the ratio of X_C to R, the increase in R is offset by the increase in X_C; hence, Q does not suffer as badly as one might expect.

8.4 Oscillator Analysis

Mathematically, if a system has poles on the $j\omega$ axis of the s plane, it forms an oscillator. That is, in the transfer function, with s replaced by $j\omega$, if there is some ω for which the denominator of the transfer function goes to zero, the system will oscillate at this frequency. In filtering terms, if the poles approach the $j\omega$ axis from the left-hand side of the s plane, the Q of the system is very high and approaches infinity. Thus, in a resonator, one can think of a negative resistance canceling the positive resistive losses to cause the Q to go to infinity, forming an oscillator. Alternatively, one can think of a feedback system, where the feedback causes the characteristic equation (the denominator of the transfer function) to go to zero for some value of $s = j\omega$. All of these ways of looking at oscillators are equivalent. Let us now look in more detail at the feedback method to analyze oscillators. Consider the feedback system shown in Figure 8.12.

The transfer function can be written as

$$\frac{v_{\text{out}}}{v_{\text{in}}} = \frac{H_1(s)}{1 - H_1(s)H_2(s)} \tag{8.3}$$

Here, the product of $H_1(s)$ and $H_2(s)$ can be seen to be the open-loop gain, often simply called the loop gain. If the open-loop gain goes to one, the transfer function goes to infinity. In the neighborhood of this singularity, the gain is very high, and a finite output voltage can be obtained for a very small input. Typically,

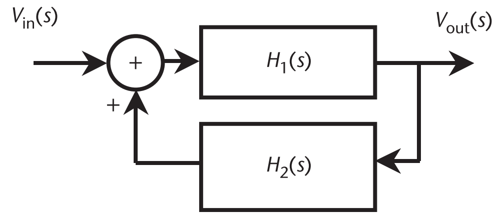


Figure 8.12 Feedback model of an oscillator.

the input to an oscillator is thermal noise. This condition is often referred to as the Barkhausen criteria, which states that the condition for oscillation is that

$$H_1(s)H_2(s) = 1 \tag{8.4}$$

This is equivalent to the following two conditions:

$$|H_1(s)H_2(s)|(s) = 1 \tag{8.5}$$
$$\angle H_1(s)H_2(s) = 0 \text{ or } 2n\pi$$

In practice, we start with a loop gain somewhat larger than one, which results in a system whose output signal grows with time. After some startup transient, the amplitude will increase to the point where nonlinearity will reduce the effective gain such that the steady-state behavior can be fairly accurately described by (8.5), with the provision that the effective large-signal g_m be used in the equations.

8.4.1 Colpitts Oscillator Analysis

To apply the above technique to the common-gate Colpitts oscillator, the small-signal model, as shown in Figure 8.13, needs to be considered. To find the loop gain, we break the loop at the source, apply a signal at v_s, and determine the gain all the way around the loop to the other side of the break at v_s'.

It can be shown that the resistor r_s across C_2 can be replaced by an equivalent resistor $r_{s,\text{tank}}$ across both capacitors according to

$$r_{s,\text{tank}} = \left(1 + \frac{C_2}{C_1}\right)^2 r_s \tag{8.6}$$

for the Colpitts common gate oscillator and

$$r_{s,\text{tank}} = \left(1 + \frac{C_1}{C_2}\right)^2 r_s \tag{8.7}$$

for the common drain oscillator. The resulting approximate transformed oscillator is seen in Figure 8.14. This model is accurate provided that the frequency of

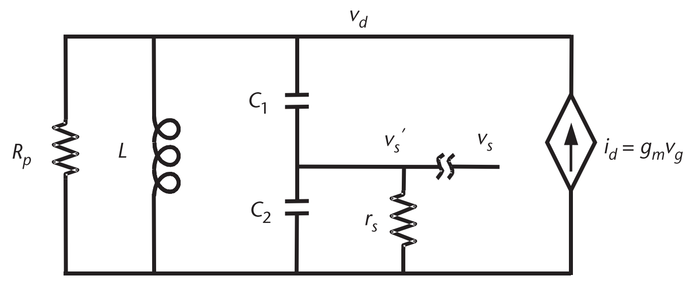


Figure 8.13 Feedback analysis of a Colpitts common-gate oscillator.

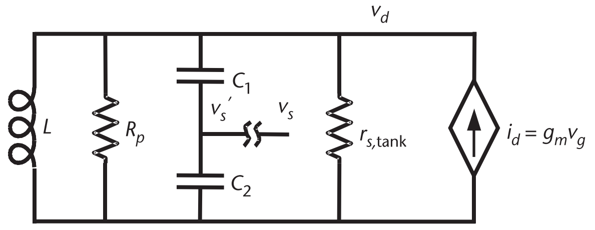


Figure 8.14 Simplified oscillator using transformation of capacitive feedback divider.

operation is well past the corner frequency of the feedback highpass filter (shown in Figure 8.13 but now missing in Figure 8.14).

The loop gain can be expressed as follows:

$$H_1H_2 = \frac{g_m}{Y_{\text{tank}}}\frac{C_1}{C_1+C_2} = \left(\frac{C_1}{C_1+C_2}\right)\cdot\frac{g_m}{\dfrac{1}{R_p}+\dfrac{1}{r_{s,\text{tank}}}+j\omega C_T-\dfrac{j}{\omega L}} \tag{8.8}$$

This can be set equal to one and solved for oscillating conditions. The imaginary terms cancel, resulting in the expected expression for the resonant frequency:

$$\omega_o = \sqrt{\frac{1}{C_T L}} \tag{8.9}$$

where C_T is the series combination of C_1 and C_2 as follows:

$$C_T = C_1 \| C_2 = \frac{C_1C_2}{C_1+C_2} \tag{8.10}$$

The remaining real terms can be used to obtain an expression for the required g_m by setting the loop gain equal to unity at the resonance frequency with the resulting expression shown in (8.11). Note that, in practice, small-signal loop gain is set to be larger than unity to guarantee startup:

$$g_m = \left(\frac{1}{R_p}+\frac{1}{r_{s,\text{tank}}}\right)\cdot\left(\frac{C_1+C_2}{C_1}\right) \tag{8.11}$$

The transconductance g_m makes up for losses in the resistors R_p and $r_{s,\text{tank}}$. Thus, for minimum required g_m, it would seem that both R_p and $r_{s,\text{tank}}$ should be made as large as possible. Large R_p is the result of high inductor Q, and large $r_{s,\text{tank}}$ is obtained by using a large value of capacitive transformer (by making C_2 bigger than C_1 in the Colpitts common-gate oscillator). However, $r_{s,\text{tank}}$ is dependent on g_m, as can be seen by noting that $r_{s,\text{tank}}$ is related to r_s by (8.7) and that r_s is the inverse of g_m. Thus, it can be shown that the required g_m is equal to

$$g_m = \frac{\omega^2 L(C_1+C_2)}{R_p} = \frac{C_1+C_2}{R_pC_T} \tag{8.12}$$

This shows that to minimize g_m, R_p should be large, but choosing a large transformer ratio (C_2/C_1) is not optimum. A large ratio of C_2 to C_1 does increase $r_{s,\text{tank}}$ but also causes the loop gain to be reduced; thus, larger g_m is required to achieve a loop gain of unity. Note that, for minimum noise, it is often advantageous to increase $r_{s,\text{tank}}$ by increasing the ratio of C_2 to C_1, but this will then require a larger g_m and, typically, a larger power dissipation. Thus, there may be a trade-off between noise and power dissipation.

8.4.2 Negative Resistance of $-G_m$ Oscillator

It is straightforward to show that a cross-coupled pair of transistors, such as those shown in Figure 8.2(c), has an input impedance of

$$Z_i = \frac{-2}{g_m} \tag{8.13}$$

Thus, in this circuit, a necessary condition for oscillation is that

$$g_m > \frac{2}{r_p} \tag{8.14}$$

where r_p is the equivalent parallel resistance of the resonator.

8.5 Amplitude of a Negative G_m Oscillator

The determination of amplitude is dependent on whether the oscillator is voltage or current limited. If the oscillator is voltage limited, typically, it is limited by the on-voltage of a switch, either to ground (for an NMOS-only oscillator) or to V_{DD} (for a PMOS-only oscillator). For an NMOS-only oscillator, the outputs are nominally at V_{DD}, with the maximum negative swing approaching ground; thus, by symmetry, the positive swing is expected to go to $2V_{\text{DD}}$ for a peak swing of V_{DD} per side or a peak swing of $2V_{\text{DD}}$ for a differential output. Similarly, for a PMOS-only oscillator, the outputs are nominally at ground potential and pulled up by the PMOS switches towards V_{DD}. This peak positive swing of V_{DD} per side is matched by a peak negative swing approaching $-V_{\text{DD}}$, again for a peak differential swing of $2V_{\text{DD}}$. For a complementary circuit with both PMOS and NMOS switches, both sides of the inductor are connected either to ground or to V_{DD}; hence, the maximum possible peak differential swing is V_{DD}. Note that bipolar transistors have a diode from base to collector; thus, bipolar negative G_m oscillators will voltage limit to $2V_{\text{BE}}$. For this reason, capacitors are often inserted between the collector and base of bipolar negative G_m oscillators to decouple the dc components. In such a case, biasing resistors need to be used to bias the gate. Since CMOS transistors do not have an equivalent diode (from gate to drain), such coupling capacitors are not used in CMOS negative G_m oscillators.

For current-limited oscillators, as long as the voltage is high enough, then the transistors can be treated as switches. Thus, each side will have current that switches between 0 and I_{bias}. A simple Fourier analysis can be used to show that the average value is $I_{\text{bias}}/2$, and the peak fundamental value is $2I_{\text{bias}}/\pi$. Thus, since the impedance per side is $R_p/2$ (current is flowing into the center tap of the inductor), the output voltage per side is

$$v_{\text{out}}\Big|_{\text{SE}} = \frac{R_p I_{\text{bias}}}{\pi} \tag{8.15}$$

and the differential voltage is

$$v_{\text{out}}\Big|_{\text{DE}} = \frac{2R_p I_{\text{bias}}}{\pi} \tag{8.16}$$

With complementary transistors, the current is flowing through the full R_p in each direction; thus, the output voltage is twice as large:

$$v_{\text{out}}\Big|_{\text{Comp\_DE}} = \frac{4R_p I_{\text{bias}}}{\pi} \tag{8.17}$$

Note that an analysis can be performed for the Colpitts oscillator as well to determine the amplitude of these circuits. Such an analysis is given in detail in [8].

In summary, typically, a specification for amplitude is given, or is indirectly given, in order to achieve a particular phase noise (since phase noise is defined as being relative to the oscillator output power). Knowing the quality of the resonator (from the Q or the parallel resistance R_p), one can determine the required bias current. Then, knowing the power-supply voltage, one can determine the total power dissipation of the oscillator.

8.6 Several Refinements to the $-G_m$ Topology

Several refinements can be made to the $-G_m$ oscillator to improve its performance. As mentioned in the previous section, for bipolar negative G_m oscillators, to increase signal swing beyond V_{BE}, the collector and base of the switching transistors can be decoupled at dc with capacitors or transformers. In either case, proper dc bias must be supplied to the base, either with resistors in the capacitively decoupled case or by providing dc to a center tap in the inductively decoupled case. Obviously, care must be taken to prevent large increases in dc power dissipation or increases in noise due to these extra components.

A modification that can be made to the $-G_m$ oscillator is to replace the high-impedance current source connected to the sources of M_1 and M_2 in Figure 8.2(c) with a resistor. Since the resistor is not a high impedance source, the bias current will vary dynamically over the cycle of the oscillation. In fact, the current will be highest when the oscillator voltage is at its peak and lowest during the zero crossings of the waveform. Since the oscillator is most sensitive to phase noise during the zero crossings, this version of the oscillator can often give very good phase noise performance. This oscillator is shown in Figure 8.15(a).

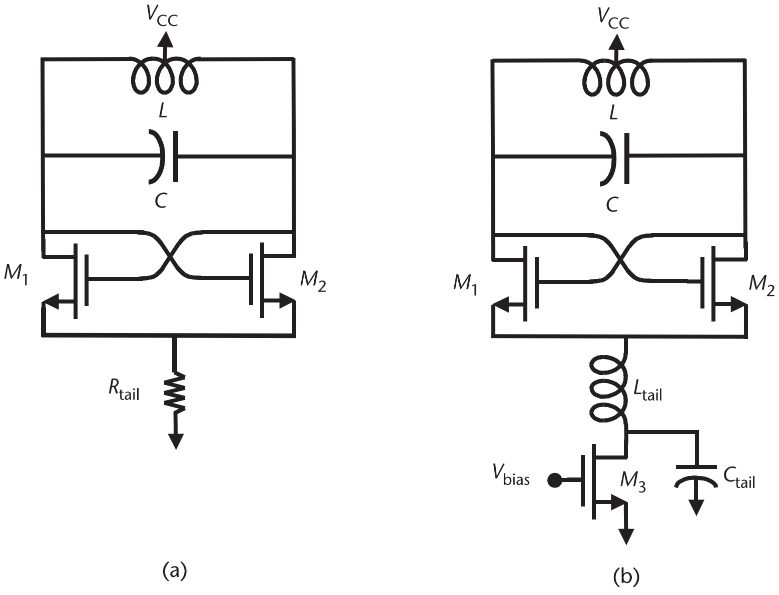


Figure 8.15 $-G_m$ oscillator with (a) resistive tail-current source, or (b) current source filter.

Since the current varies dynamically over a cycle and since the resistor R_{tail} does not require as much headroom as a current source, this allows a larger oscillation amplitude for a given power supply.

An alternative to the resistor R_{tail} is to use a noise filter in the tail, as shown in Figure 8.15(b) [9]. It is shown here with the NMOS cross-coupled negative G_m oscillator, but it could also apply to other styles as well. While the use of the inductor does require more chip area (unless it is off chip [10]), its use can lead to a very low-noise bias, leading to low phase noise designs. Noise injected by the current source around the fundamental frequency and around harmonics is filtered out. The noise implications are discussed further in Section 8.10.

Besides noise filtering, another advantage to using this filter circuit is that, before startup, the drain of transistor M_3 can be biased at a lower voltage because, during startup, the second harmonic will cause a dc bias shift at the drain of Q_3, pulling it upwards and safely away from the triode region of operation. In addition, since the second harmonic cannot pass through the inductor L_{tail}, there is no "ringing" at the drain of M_3; thus, there is a reduction of its headroom requirement. Since the inductor and capacitor are usually chosen to be large, the series resonant frequency will typically be low. Some researchers have attempted to achieve high impedance at the second harmonic of the fundamental frequency by adjusting the parallel resonance frequency between the series inductor and the parasitic capacitance on the sources of the differential pair to occur at the second harmonic. Others simply aim for a large L_{tail} and large C_{tail} to result in a better lowpass filter.

8.7 Injection-Locked Oscillators

If, rather than noise, a signal is injected into an oscillator, and if that signal is large enough, then it will pull the oscillator to that frequency. This phenomenon is known as *injection locking*. To study the effect of an injected signal, consider the model shown in Figure 8.16(a), where the oscillator feedback is shown on the left, and the injected noise and injected signal are shown on the right. The feedback transconductance g_m can be seen as providing a negative resistance $-R_n$ equal to $-1/G_m$, as shown in Figure 8.16(b). The injected noise current has an expected value of $\sqrt{F \cdot 4kT/R_p}$, where F, the equivalent noise factor of the oscillator, indicates how much additional noise is added by active circuitry. In addition, it is noted that any input resistance of the transconductance stage has been absorbed in R_p.

Under large-signal conditions, the negative and positive resistances in parallel nearly cancel out, resulting in a nearly ideal resonant circuit such that the noise input is amplified to produce the large-signal oscillator output voltage v_{out}. Since there is a finite input power, the gain cannot be infinite. However, since the gain is very large, R_n will be approximately equal to R_p. The output voltage is given by

$$v_{\text{out}} = \frac{\overline{i_n}}{\dfrac{1}{R_p} - \dfrac{1}{R_n} + sC + \dfrac{1}{sL}} = \frac{s\dfrac{\overline{i_n}}{C}}{s^2 + s\dfrac{1}{RC} + \dfrac{1}{LC}} \tag{8.18}$$

where $1/R = 1/R_p - 1/R_n$, or, equivalently, R is the parallel combination of the positive resistor R_p and the negative resistor R_n. If s is replaced by $j\omega$, the following expression results:

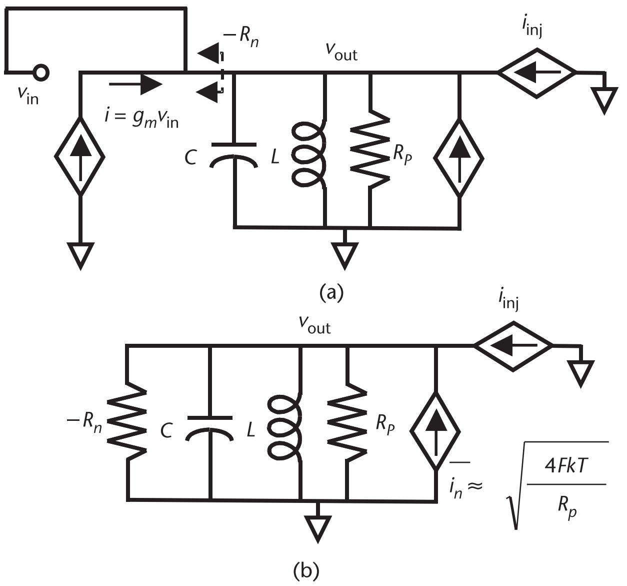


Figure 8.16 Feedback model of oscillator with noise input (this will be used to demonstrate injection locking): (a) with feedback through g_m, and (b) with $-R_n$.

$$v_{\text{out}} = \frac{\overline{i_n}}{\dfrac{1}{R} + j\left(\omega C - \dfrac{1}{\omega L}\right)} \tag{8.19}$$

The output voltage versus frequency can be seen to be a bandpass filter, as shown in Figure 8.17. Resonance occurs at ω_o according to

$$\omega_o = \frac{1}{\sqrt{LC}} \tag{8.20}$$

and, the −3-dB bandwidth is given by

$$B = \frac{1}{RC} \tag{8.21}$$

The output voltage at resonance is given by

$$v_{\text{out}} = \overline{i_n} R \tag{8.22}$$

When another signal i_{inj} is coupled into an oscillator, whether deliberately or by accident, the output will be the gain times the input signal. If the resulting

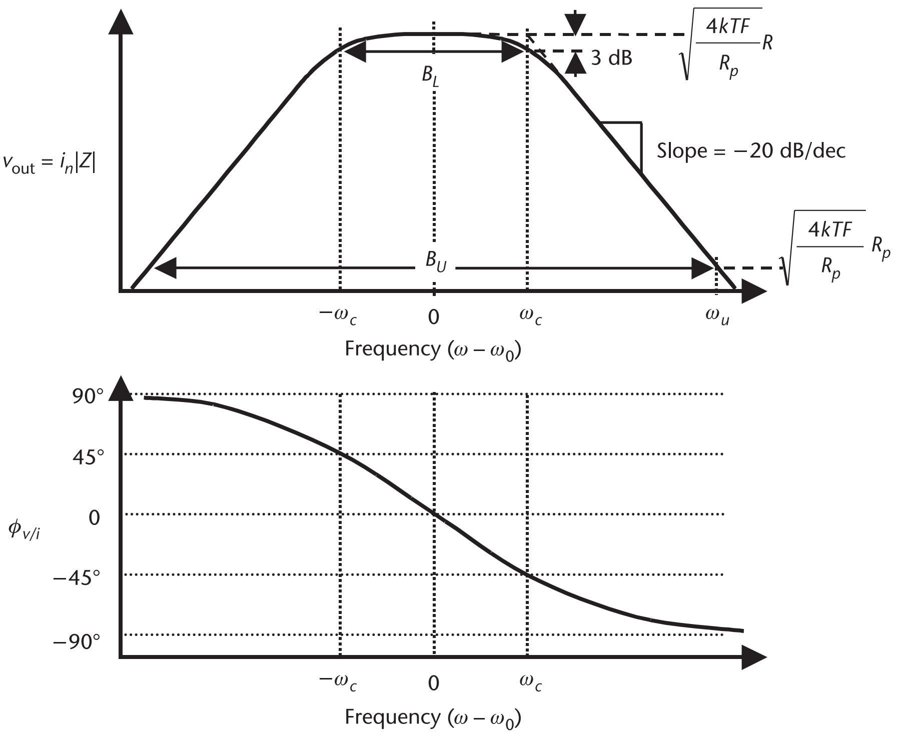


Figure 8.17 The effective gain of oscillator, assuming input is thermal noise, and the phase associated with the gain of a bandpass filter.

output signal is larger than the free-running oscillating signal, the oscillator will follow the new signal. Furthermore, in such a case, the oscillator gain will readjust itself (gain will be reduced), so the output amplitude remains approximately constant; hence, the free-running amplitude is the same as the injection-locked amplitude. This happens because of the nonlinear limiting mechanism. Note that, in general, when injection locked, since the gain is reduced, noise will be amplified by a smaller gain; hence, noise is suppressed compared to the free-running case.

To determine the condition to lock, the gain and the free-running amplitude must be determined. Figure 8.17 shows that the gain A_{osc} is simply the net impedance of the equivalent bandpass response as given by

$$A_{\text{osc}} = \frac{R}{1 + j\dfrac{\omega}{\omega_c}} = \frac{R}{1 + j\dfrac{f}{f_c}} \tag{8.23}$$

The free-running amplitude is determined by the integral under the curve of output voltage and is equal to

$$\begin{aligned}
\sqrt{v_o^2} &= \sqrt{\int_{\infty}^{\infty} |v_{\text{out}}|^2 \, d\omega} \\
&= \sqrt{\int_{\infty}^{\infty} \frac{i_n^2 R^2}{1 + \left(\dfrac{\omega}{\omega_c}\right)^2} \, d\omega} \\
&= \omega_c i_n R \sqrt{\left[\tan^{-1}\left(\frac{\omega}{\omega_c}\right)\right]_{-\infty}^{\infty}} \\
&= \overline{i_n} R \sqrt{\frac{B}{2\pi} \cdot \frac{\pi}{2}} = \frac{\overline{i_n} R \sqrt{B}}{2} \\
&= \frac{\overline{i_n}}{2} \sqrt{\frac{R}{C}}
\end{aligned} \tag{8.24}$$

It can be seen that $B \cdot \pi/2$ is the noise bandwidth of the bandpass response. Note that the equation is written for the case where noise current density is given in amperes per $\sqrt{\text{Hz}}$, and B is given in radians per second; hence, an extra factor of 2π has been included. If both terms are expressed in the same units, whether hertz or radians per second, the factor of 2π should be removed from (8.24).

The amplitude of oscillation is typically determined by the nonlinear limiting of transconductor g_m. For small signals, the value of g_m should result in a net negative resistance and an unstable circuit. With increasing amplitude, limiting causes the value of g_m to decrease until $|R_n| \approx |R_p|$ for steady-state operation. Transconductance g_m is defined by the current voltage relationship as follows:

$$i = k_1 v + k_2 v^2 + k_3 v^3 + \ldots \tag{8.25}$$

For small signals, k_1 can be seen as g_m. However, for larger signals, the third-order term will produce components at the fundamental frequency and, if k_3 is negative, will result in a decrease in the effective value of g_m. Specifically, if $v = V\cos(\omega t)$,

$$i = \left(k_1 V + \frac{3k_3 V^3}{4}\right)\cos(\omega t) + \frac{1}{2}k_2 V^2 \cos(2\omega t) + \frac{1}{4}k_3 V^3 \cos(3\omega t) + \ldots \tag{8.26}$$

Thus, the effective g_m is given by the fundamental component of $|i/v|$:

$$g_m = k_1 + \frac{3k_3 V^2}{4} \tag{8.27}$$

This can be solved for amplitude V as

$$V = \sqrt{\frac{4}{3k_3}\left(\frac{1}{R_p} - k_1\right)} \tag{8.28}$$

We note that k_1 is typically larger than $1/R_p$ to insure that the oscillator starts up. However, this means that the term in the brackets of (8.28) is negative, but, since k_3 is typically also negative, the square root can be taken. Equations (8.23) and (8.24) can be combined to determine the minimum signal amplitude for which the output due to the injected signal is equal to or larger than the output amplitude due to the noise:

$$\left|\frac{i_{\text{inj}} R}{1 + j\dfrac{f - f_o}{f_c - f_o}}\right| > \frac{\overline{i_n} R\sqrt{B}}{2} = v_o \tag{8.29}$$

Here frequencies are expressed as offsets with respect to the free-running frequency (see Figure 8.17). Note once again that the original noise input signal is a current density in amperes per $\sqrt{\text{Hz}}$, while B is in units of radians per second. For an injected signal inside of the oscillator bandwidth, the condition for lock is

$$i_{\text{inj}} > \frac{\overline{i_n}\sqrt{B}}{2} = \frac{v_o}{R} \tag{8.30}$$

For injected signals sufficiently outside the corner of the oscillator bandwidth, the condition for locking is

$$i_{\text{inj}} > \frac{\overline{i_n}(f - f_0)\sqrt{B}}{2(f_c - f_0)} = \frac{\overline{i_n}(f - f_0)\sqrt{B}}{2\dfrac{B}{2}\dfrac{1}{2\pi}} = \frac{\overline{i_n}\cdot 2\pi(f - f_0)}{\sqrt{B}} = \frac{v_o}{R}\frac{(f - f_o)}{(f_c - f_o)} \tag{8.31}$$

where $(f - f_0)$ is the frequency offset from the oscillator free-running frequency. Thus, for larger offsets, the injected signal needs to be stronger. Note that an oscillator can also injection-lock to a harmonic of a signal, and the above analysis can be used to determine the required amplitude of the harmonic signal. For an example, a free-running oscillator at 1.1 GHz can be made to lock on to the eleventh harmonic of a 100-MHz input signal, provided the input signal is nonsinusoidal enough such that it has sufficient amplitude at the eleventh harmonic. In this example, since the eleventh harmonic is close to the free-running frequency, lock will be achieved with a small input signal. Any other harmonic, being far away from the free-running frequency, will require much larger signal amplitudes to lock.

A loop analysis can be used to find an alternative expression for oscillation amplitude due to injected current i_{inj}. Input current i_{inj} can be expressed as an equivalent voltage v_{inj} applied at the input of the transconductor by dividing current i_{inj} by g_m. Then, the output voltage is found from the forward gain FG in terms of the transconductance g_m and the open-loop conductance Y_{OL} as follows:

$$\begin{aligned} v_{\text{out}} &= v_{\text{inj}} \frac{\text{FG}}{1 - \text{FG}} = \frac{i_{\text{inj}}}{g_m} \frac{\dfrac{g_m}{Y_{\text{OL}}}}{1 - \dfrac{g_m}{Y_{\text{OL}}}} = i_{\text{inj}} \frac{1}{Y_{\text{OL}} - g_m} \\ &= \frac{i_{\text{inj}}}{\left\{\dfrac{1}{R_p} + j\omega C\left[1 - \left(\dfrac{\omega_o}{\omega}\right)^2\right]\right\} - \dfrac{1}{R_n}} \end{aligned} \tag{8.32}$$

Not surprisingly, the result can be shown to be the same as previously shown in (8.18). We can manipulate the expression for Y_{OL} by noting that the unloaded quality factor Q_U is equal to ω_o/B, and bandwidth B is equal to $1/R_pC$. As a result, the following expression for open-loop admittance Y_{OL} can be obtained:

$$Y_{\text{OL}} = \frac{1}{R_p} + j\omega C\left[1 - \left(\frac{\omega_o}{\omega}\right)^2\right] = \frac{1}{R_p}\left\{1 + jQ_U \frac{\omega}{\omega_o}\left[1 - \left(\frac{\omega_o}{\omega}\right)^2\right]\right\} \tag{8.33}$$

If ω is replaced by $\omega_o + \Delta\omega$, and the approximation is made that $\Delta\omega$ is much less than ω_o, the following approximation is obtained:

$$Y_{\text{OL}} = \frac{1}{R_p}\left\{1 + jQ_U \frac{\omega}{\omega_o}\left[1 - \left(\frac{\omega_o}{\omega_o + \Delta\omega}\right)^2\right]\right\} \approx \frac{1}{R_p}\left(1 + j2Q_U \frac{\Delta\omega}{\omega_o}\right) \tag{8.34}$$

Substituting back into the original expression,

$$v_{\text{out}} = i_{\text{inj}} \frac{1}{Y_{\text{OL}} - g_m} \approx \frac{i_{\text{inj}}}{\dfrac{1}{R_p}\left(1 + j2Q_U \dfrac{\Delta\omega}{\omega_o}\right) - g_m} = \frac{i_{\text{inj}} R_p}{1 - g_m R_p + j2Q_U \dfrac{\Delta\omega}{\omega_o}} \tag{8.35}$$

It is interesting to note that this has been interpreted to mean that the output voltage depends on the original parallel resistance and, hence, the Q of the tank circuit. However, R_p and Q_U can both be eliminated from this expression by noting that since $|R_p|$ is approximately equal to $|R_n|$, $1 - g_m R_p$ is very small. Thus, the expression for output voltage (which could have been derived simply enough directly from the original expression) is given by

$$v_{\text{out}} \approx \frac{i_{\text{inj}}}{j2C\Delta\omega} \tag{8.36}$$

This very simple result, which can be used as an alternative to (8.31), seems to show that the original Q_U and parallel resistance R_p are irrelevant. This can be explained by noting that feedback produces negative resistance, which exactly cancels the original positive resistance. However, one should not be too hasty in stating that original R_p and Q_p are irrelevant since the noise current i_n has a component directly related to R_p or, equivalently, Q_U, and, depending on the original formulation of the equivalent circuit, the factor of g_m can appear in the expression for v_{out}.

Example 8.1: Injection Locking

Demonstrate injection locking with a simple model of a transconductor.

Solution: The injection-locking circuit is as shown in Figure 8.16. We start by choosing an inductance and capacitance of 5 nH and 5.06 pF to result in an oscillating frequency close to 1 GHz. The calculated frequency using (8.20) is predicted to be 1.0006 GHz. Note that there is a 600-kHz offset from 1 GHz. With a Q of 10, the inductor has a parallel resistance of about 314Ω. Noise injection due to the resistor is

$$i_n = \sqrt{\frac{4kT}{R}} = \sqrt{\frac{4 \times 1.38 \times 10^{-23}\ \text{J/K} \times 300\text{K}}{314\Omega}} = 7.26\ \text{pA}/\sqrt{\text{Hz}}$$

To represent noise inputs, 40 current sources are placed in parallel, separated by 100 kHz from 0.98 GHz to 1.02 GHz, and each has a current amplitude of 2 nA, representing the noise from a 314Ω resistor in a 100-kHz bandwidth. The transconductor is modeled with a voltage-controlled current source with transfer function:

$$i_o = 0.005v_i - 0.0005v_i^3$$

The third-order term is sufficient to produce g_m compression and amplitude limitation. The simulated free-running frequency is 1.0003 GHz, as shown in Figure 8.18. Note that the time step is 5 ps, and total simulation time is 40 ms for a resolution of 25 kHz. From (8.28), the amplitude is predicted to be

$$V = \sqrt{\frac{4}{3k_3}\left(\frac{1}{R_p} - k_1\right)} = \sqrt{\frac{4}{3 \cdot (-0.0005)}\left(\frac{1}{314} - 0.005\right)} = 2.20\text{V}$$

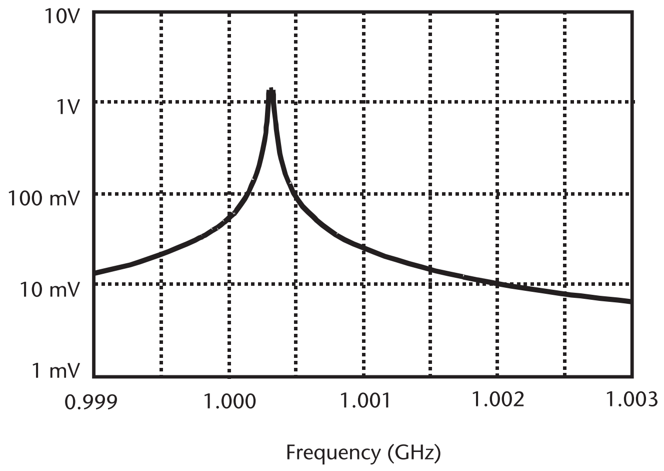


Figure 8.18 Free-running frequency of the oscillator model.

Time domain plots show an amplitude of 2.1V, in agreement with the prediction and with the spectrum shown in Figure 8.18. Note that the frequency is not all in one frequency bin; thus, the total power must be taken to derive the amplitude from the spectrum. In this case, two bins are dominant, with about 1.4V each, for a total of just over 2V, in agreement with the time domain simulation.

Thus, the effective g_m can be determined from (8.27):

$$\begin{aligned} g_m &= k_1 + \frac{3k_3V^2}{4} \\ &= 0.005 + \frac{3 \cdot (-0.0005) \cdot 2.1^2}{4} \\ &= 0.005 - 0.001654 \\ &= 0.003346 \end{aligned}$$

The effective value of g_m is 3.346 mA/V, which is down from the small-signal value of 5 mA/V. With these values, for an input signal injected at 1 GHz, assuming this is outside of the oscillator bandwidth, (8.36) predicts that the required injection current i_{inj} for locking is

$$i_{\text{inj}} > v_{\text{out}} \cdot 2C\Delta\omega = 2.2 \times 2 \times 5.06p \times 2\pi \times 300k = 41.95\ \mu\text{A}$$

This can also be determined from (8.31), but to use this equation, R and B must be known. Equivalent closed-loop parallel resistance R can be determined from (8.24) as

$$R = \frac{4v_o^2C}{\overline{i_n^2}} = \frac{4 \times 2.1^2 \times 5.06p}{\left(\frac{2n}{\sqrt{100k}}\right)^2} = 2.2315 \times 10^{12}\,\Omega$$

Bandwidth is determined to be

$$B = \frac{1}{RC} = \frac{1}{2.2315 \times 10^{12}\Omega \times 5.06p} = 0.88565 \text{ rad/s or } 0.014096 \text{ Hz}$$

Then, the required current for injection can be found from (8.31):

$$i_{\text{inj}} > \frac{v_{\text{out}}}{R}\frac{(f - f_o)}{(f_c - f_o)} = \frac{2.2 \times 300k}{2.2315 \times 10^{12} \times 0.014096/2} = 41.96\ \mu\text{A}$$

This is in agreement with the previous calculation. These calculations also demonstrate the extremely high resistance and narrow bandwidth of oscillator circuits. A series of injected tones is applied, with the results shown in Figure 8.19. The simulated current required for locking is 44 μA, just slightly higher than the predicted current.

8.7.1 Phase Shift of Injection-Locked Oscillator

From the model of the oscillator in Figure 8.16, i_{total}, the total current injected into the resonant circuit, is the vector sum of i and i_{inj}. If the oscillating frequency

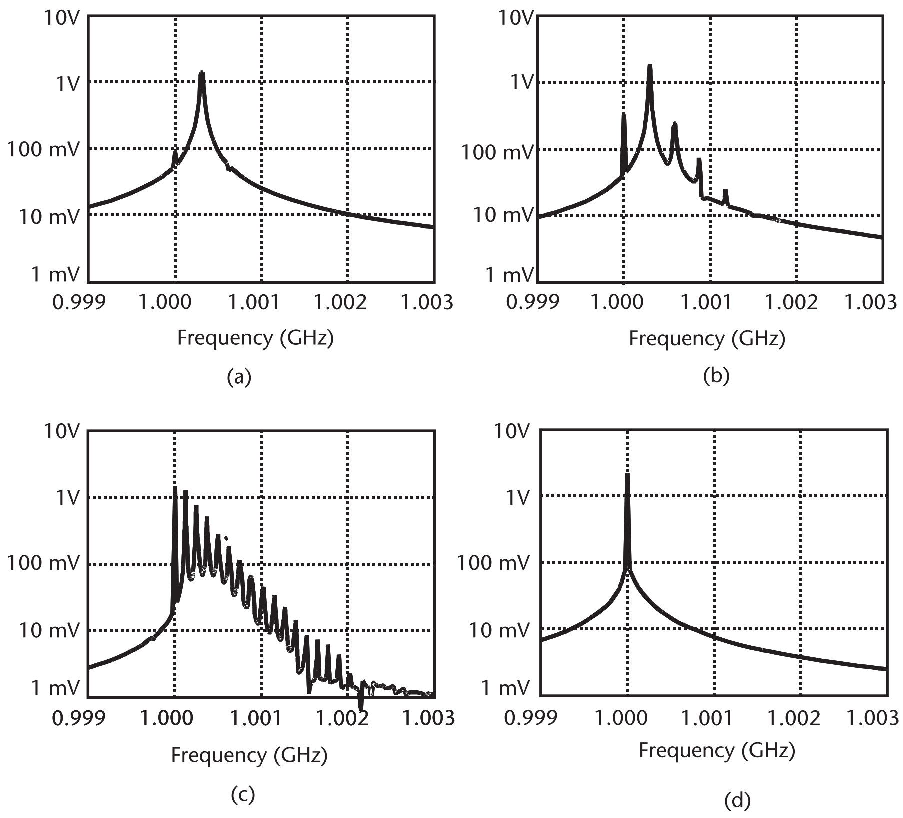


Figure 8.19 Oscillator model with increasing injected current at 1 GHz: (a) 2 μA (nearly identical to free running), (b) 15 μA, (c) 40 μA, and (d) 44 μA (injection locked).

is not at the center frequency of the bandpass filter, there will be phase shift between the total current injected into the tank and the voltage it produces as given by

$$\phi_{\text{osc}} = -\tan^{-1}\left[\left(\omega C - \frac{1}{\omega L}\right)R\right] \tag{8.37}$$

As noted before, the value of R is ultimately set by the nonlinear limiting mechanism, typically the transconductor. With an injected signal, the output voltage is still determined from the integral as in (8.24), however, with the additional term of $i_{\text{inj}}R$ added to it. Since the output voltage remains roughly constant, it is clear that for larger injected signals, the value of R must decrease; hence, the gain to the noise also decreases. Also, as seen from (8.37), with a decrease in R, there is less phase shift for a given frequency offset.

The phase in (8.37) is the total phase and is the combination of two components. The first is the phase between i and v. Since i is directly created from the voltage, it must be in phase with the voltage. Hence, there must be additional phase shift between i_{inj} and the voltage such that the total current has the correct phase shift with respect to the voltage, as shown in Figure 8.20. In the general case, the phase shift can be calculated from

$$\phi_{\text{inj}} = \sin^{-1}\left(\frac{i_{\text{total}}}{i_{\text{inj}}}\sin\phi_{\text{osc}}\right) \tag{8.38}$$

Note that if the amplitudes of current are the same (as will be the case in the quadrature oscillator discussed in Section 8.8), then the phase shift ϕ_{inj} will be twice that required by the bandpass filter ϕ_{osc}.

$$\phi_{\text{inj}} = 2\phi_{\text{osc}} = -2\tan^{-1}\left[\left(\omega C - \frac{1}{\omega L}\right)R\right] \tag{8.39}$$

In the special case where phase is 90°, the filter has phase shift of 45° and occurs at

$$\omega_{90^\circ} = \sqrt{\omega_0^2 + \frac{B^2}{4}} \pm \frac{B}{2} \approx \omega_0 \pm \frac{B}{2} \tag{8.40}$$

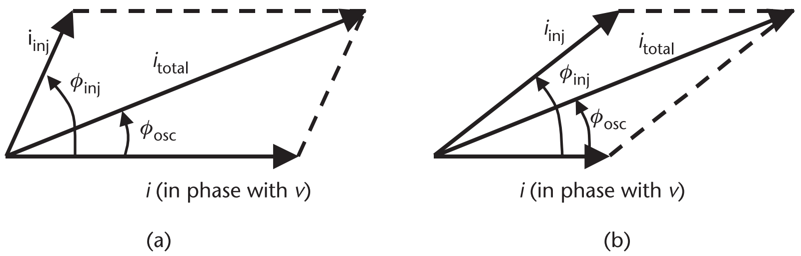


Figure 8.20 Phase shift of an external signal i_{inj} with respect to the output voltage: (a) for a small injected signal, and (b) for a larger injected signal.

where the approximation is valid if the center frequency is much larger than the bandwidth.

Note that, in practice, deciding what value of R to use in the above equations to predict phase shift is not obvious, since R changes with the amount of injected current. From experience, and somewhat surprisingly, it has been observed that using the parallel tank resistor R_p for R gives good results. This is demonstrated in Example 8.2.

Example 8.2: Phase Shift with Injection Locking

As a continuation of Example 8.1, predict and simulate the phase shift for an injection current of 100 μA. Also compare this to the phase shift when i_{inj} and i are equal.

Solution: With an oscillating voltage of 2.1V, as determined in Example 8.1, total current is found by multiplying by g_m the feedback transconductance, found in Example 8.1 to be 3.346 mA/V. Thus, total current is 7.03 mA and phase shift can be predicted to be

$$\phi_{\text{inj}} = \sin^{-1}\left(\frac{i_{\text{total}}}{i_{\text{inj}}}\sin\,\phi_{\text{osc}}\right) = \sin^{-1}\left(\frac{7.03\text{ mA}}{0.1\text{ mA}}\sin\left\{-\tan^{-1}\left[\left(\omega C - \frac{1}{\omega L}\right)R\right]\right\}\right)$$

This is compared to simulations with results in Figure 8.21. As discussed above, R is set equal to R_p (314Ω) with the result that there is near perfect agreement between simulations and theory. As can be seen, at 1.0003 GHz, theoretical and simulated phase is 0°. Note that 1.0003 GHz is the free-running frequency and is also equal to the resonant frequency of the parallel tank circuit. As seen in the figure, when phase shift reaches approximately 90°, the oscillator is no longer locked. For comparison, the curves for 50 μA and 200 μA are also shown, verifying that, with more current, the phase shift is lower.

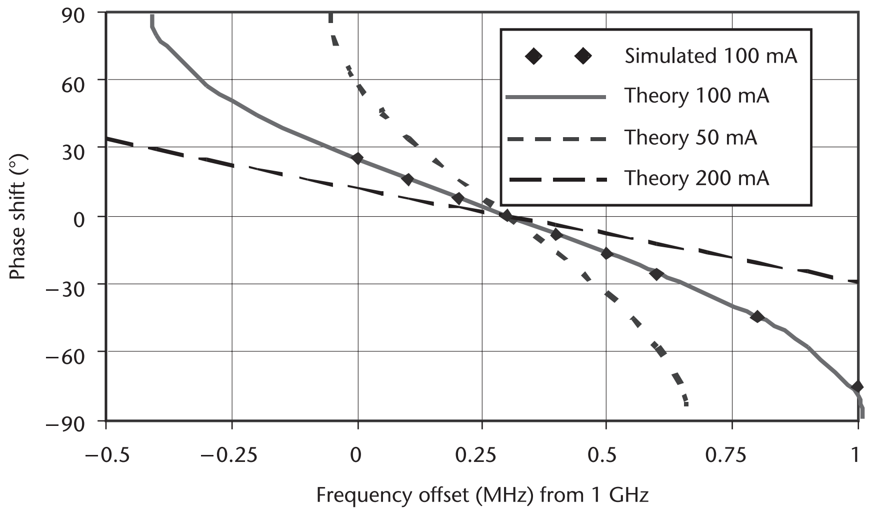


Figure 8.21 Theoretical and simulated phase shift between an injected signal and a locked output tone at 100 μA of injected current. The theoretical phase shift is also shown for an injected signal at 50 μA and 200 μA.

The phase shift can also be explored as a function of the magnitude of the injected signal with the input frequency at a constant 1.002 GHz. The results are shown in Figure 8.22. This figure shows that with larger injected amplitude, phase shift is decreased. It was observed from both theory and simulation that the minimum current for injection locking at this offset frequency is about 240 μA.

To verify that the system is not locked for injected currents less than 240 μA, 235 μA is injected, and the resulting phase plotted as a function of time is shown in Figure 8.23. This shows the obvious cycle slipping, verifying that the oscillator is not locked.

8.8 Quadrature LC Oscillators Using Injection Locking

Two oscillators can be connected in such a way that a signal from one oscillator is injected into the second oscillator and a signal from the second oscillator is injected into the first. The result is that the two oscillators become locked in

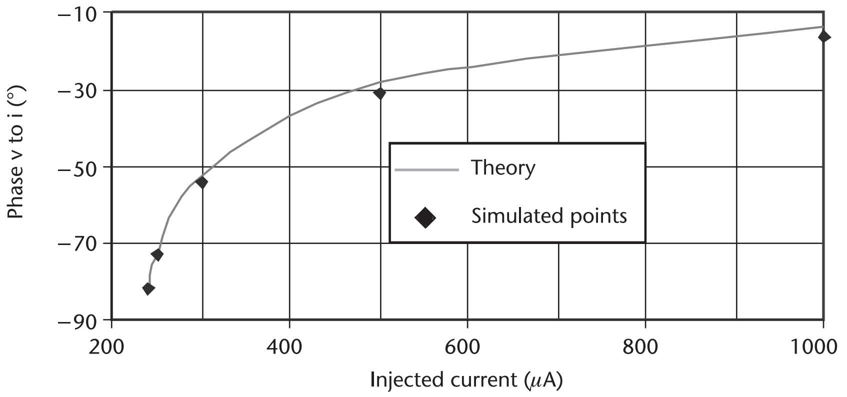


Figure 8.22 Theoretical and simulated phase shift between an injected signal and a locked output tone for current injected at 1.002 GHz, or about 1.7 MHz offset from the free-running frequency. Lock is achieved only for currents larger than about 240 μA. Predicted locking current is 227 μA.

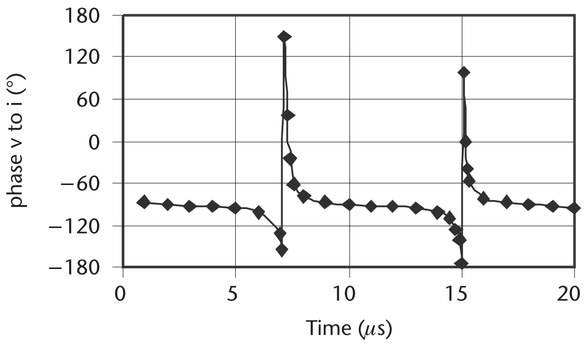


Figure 8.23 The phase shift between an injected signal and a locked output tone for current injected at 1.002 GHz at 235 μA.

frequency, typically oscillating in quadrature. Very often, quadrature signals are required; for example, mixers in an image reject configuration require quadrature oscillator signals. Other techniques to generate quadrature signals, including polyphase filters and ring oscillators, are discussed in Section 8.9.

8.8.1 Parallel Coupled Quadrature LC Oscillators

The most common technique is the parallel connection shown in Figure 8.24, where each oscillator is made up of a tank circuit and cross-coupled feedback circuit. In addition, each oscillator output is connected to the other oscillator with transistors in parallel to the cross-coupled transistors. Thus, oscillator 1 has feedback transistors M_1 and M_2 and coupling from oscillator 2 via transistors M_5 and M_6. Typically, feedback and coupling transistors are made the same size. Furthermore, because of symmetry, the oscillation amplitudes of the two oscillators should be the same.

To understand why the two oscillators oscillate in quadrature, first note that the two oscillators can be modeled as gain stages, as shown in Figure 8.25. Each

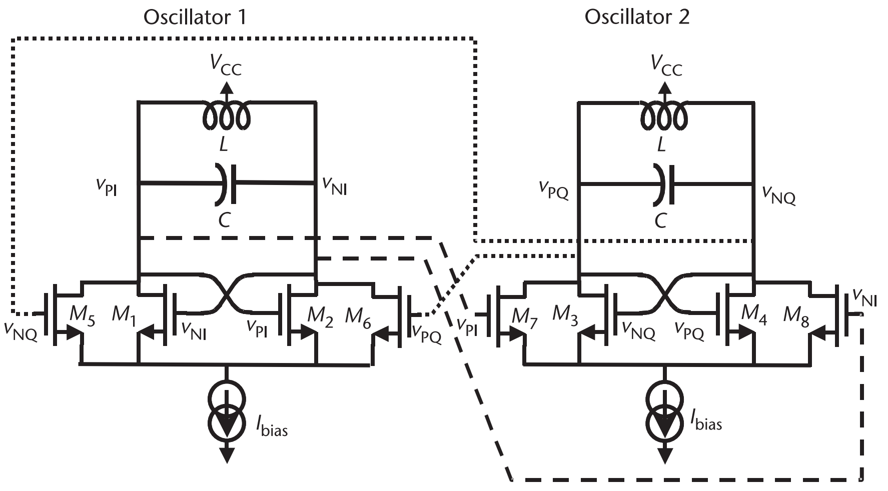


Figure 8.24 Quadrature negative G_m oscillator with parallel cross connections.

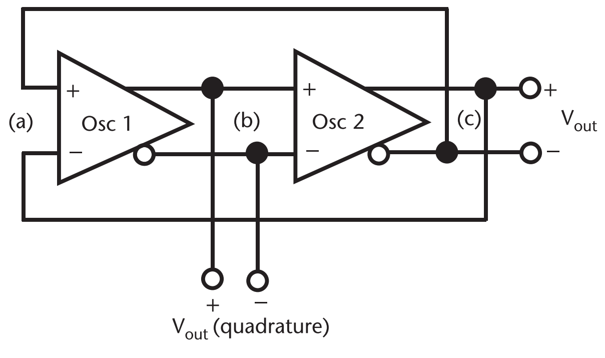


Figure 8.25 A quadrature oscillator modeled as two amplifier stages in feedback.

stage has a gain and a phase, as shown in Figure 8.17. We note that, as for any oscillator structure, the loop phase must be 0° or 360°. Thus, since the crossed wires at the output represent a phase shift of 180°, the two oscillators together must have an additional phase shift of 180°. Hence, if all components are matched, the phase shift across each oscillator will be 90°. We note that Figure 8.17 shows the transfer function to the total current injected into the tank and that, for such current, the phase only asymptotically approaches 90°. The total current is made up of a combination of two equal currents, as shown in Figure 8.26.

The first current, i, is locally fed back in direct proportion to the voltage; hence, it is in phase with the voltage. The second component, i_{inj}, comes from the other oscillator. Since the two currents have the same amplitude, for the two oscillators to be 90° out of phase, the total current is 45° out of phase with respect to the voltage. By this argument, it can be seen that the frequency of operation will be shifted from the resonant frequency by an amount of $B_L/2$, as shown in Figure 8.17, since this is where there is 45° of phase shift.

Phase shift in the injection-locked oscillator was previously given by (8.39), which is repeated here, and is derived with equal currents for feedback and cross coupling.

$$\phi_{\text{inj}} = 2\phi_{\text{osc}} = -2\tan^{-1}\left[\left(\omega C - \frac{1}{\omega L}\right)R\right]$$

In the special case where phase is 90°, the filter has phase shift of 45° and occurs at

$$\omega_{90°} = \sqrt{\omega_0^2 + \frac{B^2}{4}} \pm \frac{B}{2} \approx \omega_0 \pm \frac{B}{2} \tag{8.41}$$

where the approximation is valid if the center frequency is much larger than the bandwidth. In practice, these equations work well if R_p is used for R.

An analysis of the loop gain of the quadrature oscillator model in Figure 8.27 results in

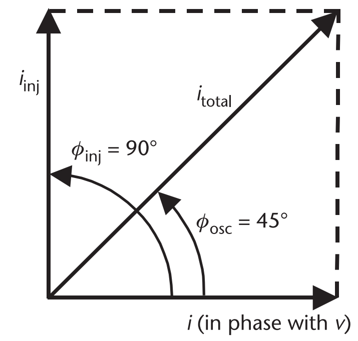


Figure 8.26 The current injected into each resonator for a quadrature oscillator.

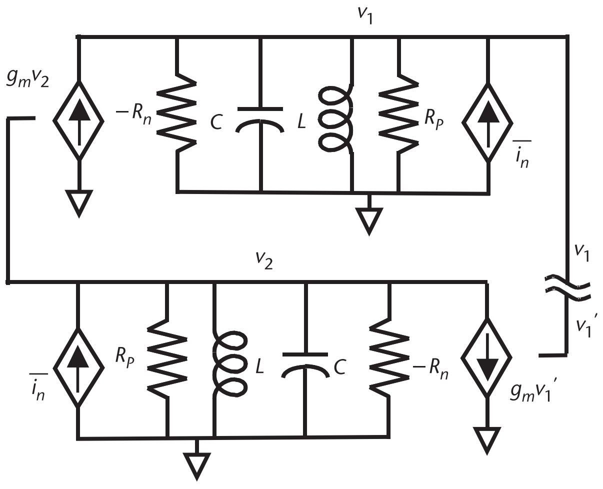


Figure 8.27 A model for a quadrature oscillator.

$$\frac{v_1}{v_1'} = \frac{-g_m^2}{\left[\left(g_m - \frac{1}{R}\right) + j\left(\omega C - \frac{1}{\omega L}\right)\right]^2} \tag{8.42}$$

If it is assumed that each oscillator in steady state has adjusted its g_m such that it is equal (or close) to the value of $1/R$, and if the loop gain is set equal to 1, then,

$$\omega^2 LC \pm \omega L g_m - 1 = 0 \tag{8.43}$$

The "±" of the middle term is used because one can find two solutions. One solution is below the resonant frequency, and one is above the resonant frequency. The solution for ω is

$$\omega = \sqrt{\frac{1}{LC} + \frac{g_m^2}{4C^2}} \pm \frac{g_m}{2C} \approx \omega_0 \pm \frac{B}{2} \tag{8.44}$$

where B is the equivalent bandwidth and is given by g_m/C.

Thus, although some approximations have been made, the result is that the circuit behaves like two coupled, parallel, resonant circuits with equivalent bandwidth of g_m/C. If it is noted that g_m is nominally equal to $1/R_p$, then the frequency of oscillation is in exact agreement with the previous determination. However, the phase shift does not agree exactly at frequencies other than the resonant frequency, but if such phase information is desired, (8.39) can be used.

Example 8.3: Quadrature Oscillator Design

Design a 1-GHz quadrature oscillator using simplified models to demonstrate phase shift and amplitude theory. Change the capacitance by 1%, and observe and explain the resulting amplitude and phase mismatch.

Solution: A 1-GHz quadrature oscillator is built with a resonant tank made up of 5 nH and 5.06 pF. Feedback and cross-coupling transconductors have transfer functions $i = 0.005v - 0.0005v^3$. Initially, the circuit is run open loop with a voltage representing VCO_2, with that voltage adjusted to equal the oscillating amplitude of VCO_1, and magnitude and phase results are obtained as in Figure 8.28.

Thus, where amplitude is rolled off by 3 dB (down to about 2.1V), phase is at 90°, as is expected from the explanation around Figure 8.26. When connected as an injection-locked oscillator, the oscillating frequency is 1.05144 GHz, at the frequency where the phase shift is 90° with an amplitude of about 2.2V, also as expected. The phase shift between the two oscillators is exactly 90° within the simulation limits (better than a thousandth of a degree). When capacitance of one oscillator is increased by 1%, the frequency decreases to 1.0487 GHz, and the phase shift is now 93.93°. This can be explained by examining a zoom in of the phase versus frequency plot derived from (8.39) and shown in Figure 8.29. When both capacitors are at 5.06 pF, each phase shift is 90° at a predicted frequency of 1.0519 GHz, close to the simulated frequency of 1.05144 GHz. When one capacitor is high by a fraction δ (in this case, 1%) at 5.1106 pF, its resonant frequency

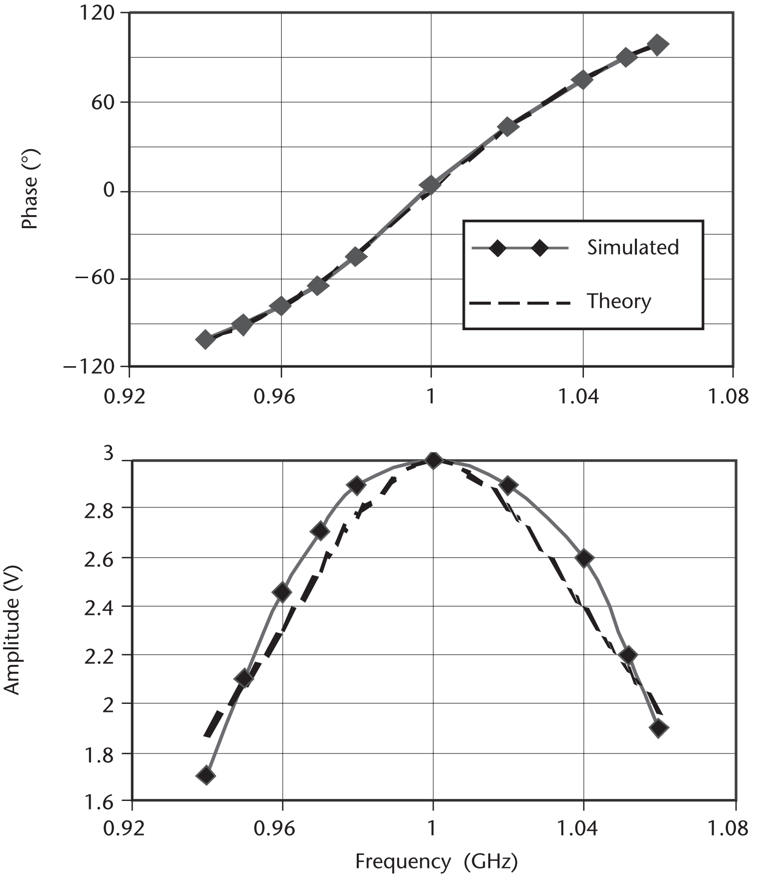


Figure 8.28 Quadrature oscillation (a) phase and (b) amplitude for an injected signal (open-loop simulation).

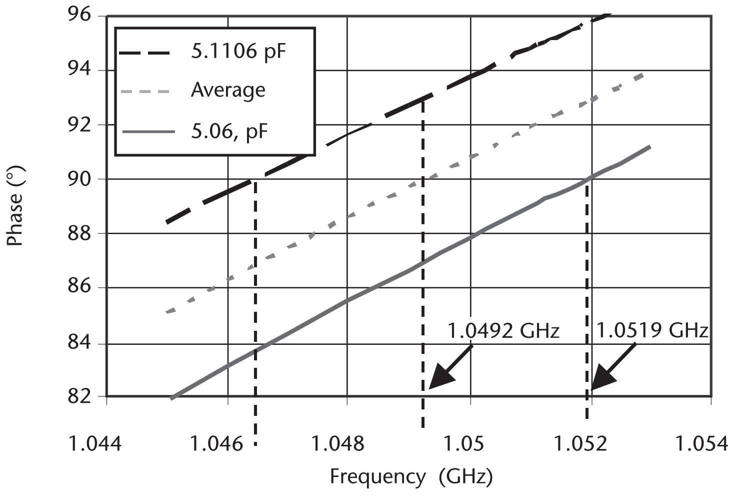


Figure 8.29 Phase to an injected signal for a quadrature oscillator with nominal capacitor of 5.06 pF and 1% increased capacitor of 5.1106 pF (open-loop simulation).

decreases by about $\delta/2$ (in this case, by 0.5%); consequently, there is more phase shift at the frequency of interest. The total phase still has to be 90°; hence, frequencies adjusts themselves until the sum of the two phase shifts is 180°. This new frequency can be found by noting where the average of the two phase shifts goes through approximately 90° at a frequency shift of $\delta/4$ (or 0.25%). From the starting frequency of 1.0519 GHz, a 0.25% shift will move it to 1.0493 GHz, while the equation and Figure 8.29 predict a new frequency of 1.0492 GHz. Both are close to the simulated frequency of 1.0487 GHz. Also of importance, the phase shift across the two oscillators is now seen to be about 87° and 93° for a total phase of 180°, again in agreement with the simulations. Phase shift can be shown to be related to bandwidth by

$$\left|\frac{d\phi}{d\omega}\right| = \frac{2Q}{\omega_o} = \frac{2}{B} \tag{8.45}$$

Hence, the phase shift is estimated at

$$\Delta\phi = \frac{\Delta\omega}{B/2} = \frac{\omega_{\text{matched}} \cdot \delta/4}{B/2} \tag{8.46}$$

where δ is the capacitor mismatch (0.01), ω_{matched} is the quadrature oscillator frequency with components matched ($2\pi \times$ 1.0519 GHz), and $B/2$ is the difference between the LC resonant frequency and the free-running frequency ($2\pi \times$ 51.6 MHz). This equation predicts a phase offset of 2.92°, which is quite close to the value predicted from Figure 8.29. The simulated phase change is slightly larger at 3.93° but still illustrates the usefulness of this estimate.

Similarly, the change of phase shift is also related to a change of amplitude, as seen in Figure 8.28. Simulated results show the amplitudes are now 2.08V and 2.35V, in agreement with the above explanation.

8.8.2 Series Coupled Quadrature Oscillators

Quadrature oscillators can also have series coupling, as shown in Figure 8.30. These two topologies have been studied and compared by [11]. A conclusion drawn was that optimal coupling for the parallel circuit results when the coupling and main transistor are approximately the same size. For the series circuit, the coupling transistors should be about five times bigger than the main transistors. For optimal coupling, the parallel circuit appears to have better quadrature phase matching; however, if its coupling transistor size is adjusted so that the quadrature phase matching is equal, then the series coupled circuit is stated to have 10 dB to 20 dB better phase noise.

8.8.3 Other Quadrature-Generation Techniques

All of the quadrature schemes shown so far have some drawbacks. For example, the series coupled oscillator requires a coupling transistor about five times bigger than the other transistors. These coupling transistors will have significant parasitic capacitance, which will limit tuning range. The parallel coupled circuit has additional power dissipation due to the coupling transistors and has less than ideal phase noise. In both of the above coupling techniques, since the injected signal is being applied at 90° to the oscillating signal, the peak injected current occurs at the zero crossing of the oscillator signal, and, at this point, the oscillator is most sensitive to injected noise. It has been shown in [12] and elsewhere that, by applying 90° of phase shift in the coupling path, the injected signal can be in phase with the oscillating signal, and phase noise is substantially improved. However, this improvement comes at the cost of needing further resonator circuits and does not

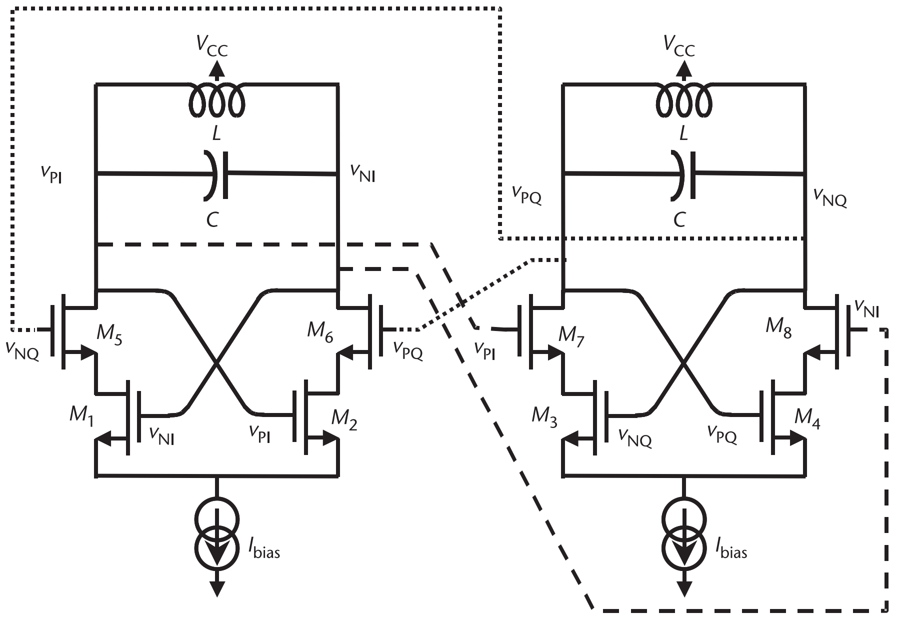


Figure 8.30 A quadrature negative G_m oscillator with series cross connections.

solve the other problems noted above (increased power dissipation for parallel coupling and reduced tuning range for series coupling). An alternative way to improve phase noise is to couple the two oscillators by use of the voltage on the current source node. Since this voltage is typically at a harmonic of the oscillating frequency, this technique is referred to as superharmonic coupling [13] and is shown in Figure 8.31. This technique is lower power than the parallel technique as no extra current paths are required.

8.9 Other Techniques to Generate Quadrature Signals

There are other ways to generate quadrature signals. One common technique is for an oscillator to be followed by a polyphase filter [8]. More than one filtering stage can be used if one desires more phase accuracy and operation over a wider bandwidth. Polyphase filters consume some additional power due to the buffers required. Except for the buffers, they are passive RC circuits, and, as a result, there is typically a 3-dB loss of signal amplitude per filtering stage. Another way to generate quadrature signals is to design an oscillator to run at twice the desired frequency, then to use a divider to generate the quadrature outputs at the desired frequency. The disadvantage is the additional complexity and power dissipation inherent in oscillators operating at twice the frequency. Another technique to generate quadrature signals, which will be discussed in Section 8.12, is the use of differential ring oscillators with two or four stages.

8.10 Phase Noise in LC Oscillators

A major challenge in most oscillator designs is to meet the phase noise requirements of the system. An ideal oscillator has a frequency response that is a simple impulse

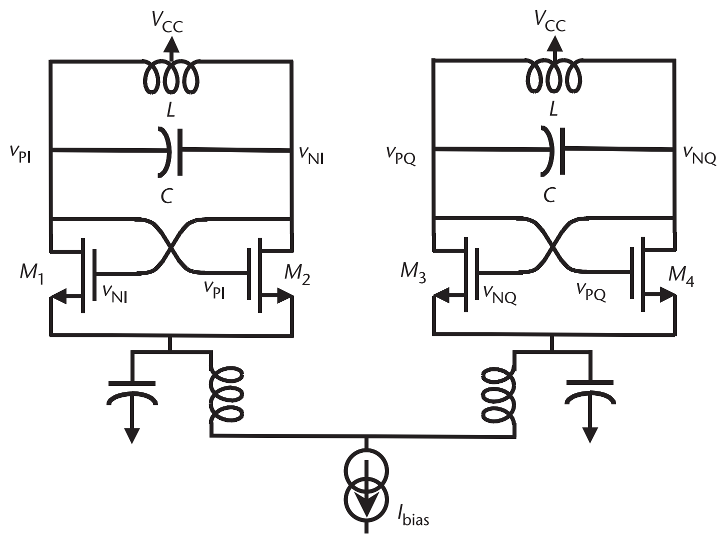


Figure 8.31 A quadrature negative G_m oscillator using superharmonic coupling.

at the frequency of oscillation. However, real oscillators exhibit "skirts" caused by instantaneous jitter in the phase of the waveform. Noise that causes variations in the phase of the signal (distinct from noise that causes fluctuations in the amplitude of the signal) is referred to as *phase noise*. The waveform of a real oscillator can be written as

$$V_{\mathrm{OSC}} = A\cos[\omega_o t + \phi_n(t)] \tag{8.47}$$

where $\phi_n(t)$ is the phase noise of the oscillator. Here, amplitude noise is ignored because it is usually of little importance in most system specifications. Because of amplitude limiting in integrated oscillators, typically AM noise is lower than FM noise. There are several major sources of phase noise in an oscillator, and they will be discussed next.

8.10.1 Linear or Additive Phase Noise and Leeson's Formula

In order to derive a formula for phase noise in an oscillator, we will start with the feedback model of an oscillator as shown in Figure 8.32. For the purpose of this analysis, we will assume that H_1 is equal to one; hence, we can use the circuit shown in Figure 8.32(b).

From control theory, it is known that

$$\frac{N_{\mathrm{OUT}}(s)}{N_{\mathrm{IN}}(s)} = \frac{1}{1 - H(s)} \tag{8.48}$$

where $H(s) = H_1(s)H_2(s)$. $H(s)$ can be written as a truncated Taylor series:

$$H(j\omega) \approx H(j\omega_o) + \Delta\omega\frac{dH}{d\omega} \tag{8.49}$$

Since the conditions of stable oscillation must be satisfied, $H(j\omega_o) = 1$. Now (8.48) can be rewritten, using (8.49), as

$$\frac{N_{\mathrm{OUT}}(s)}{N_{\mathrm{IN}}(s)} = \frac{1}{-\Delta\omega\dfrac{dH}{d\omega}} \tag{8.50}$$

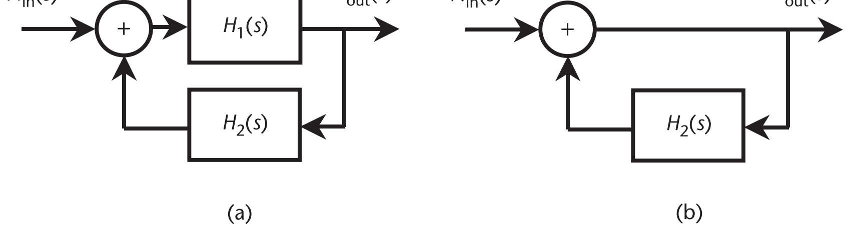


Figure 8.32 Feedback model of oscillator: (a) with a feedforward gain of H_1, and (b) with a feedforward gain of unity.

Noise power is of interest here, so

$$\left|\frac{N_{\mathrm{OUT}}(s)}{N_{\mathrm{IN}}(s)}\right|^2 = \frac{1}{(\Delta\omega)^2\left|\frac{dH}{d\omega}\right|^2} \tag{8.51}$$

This equation can now be rewritten using $H(\omega) = |H|e^{j\phi}$ and the product rule:

$$\frac{dH}{d\omega} = \frac{d|H|}{d\omega}e^{j\phi} + |H|je^{j\phi}\frac{d\phi}{d\omega} \tag{8.52}$$

Noting that the two terms on the right of (8.52) are orthogonal,

$$\left|\frac{dH}{d\omega}\right|^2 = \left|\frac{d|H|}{d\omega}\right|^2 + |H|^2\left|\frac{d\phi}{d\omega}\right|^2 \tag{8.53}$$

At resonance, phase changes much more quickly than magnitude, and $|H| \approx 1$ near resonance. Thus, the second term on the right is dominant, and this equation reduces to

$$\left|\frac{dH}{d\omega}\right|^2 = \left|\frac{d\phi}{d\omega}\right|^2 \tag{8.54}$$

Now, substituting (8.54) back into (8.51),

$$\left|\frac{N_{\mathrm{OUT}}(s)}{N_{\mathrm{IN}}(s)}\right|^2 = \frac{1}{(\Delta\omega)^2\left|\frac{d\phi}{d\omega}\right|^2} \tag{8.55}$$

By noting that the rate of change of phase can be related to the Q by

$$Q = \frac{\omega_o}{2}\left|\frac{d\phi}{d\omega}\right| \tag{8.56}$$

Equation (8.55) can be rewritten as

$$\left|\frac{N_{\mathrm{OUT}}(s)}{N_{\mathrm{IN}}(s)}\right|^2 = \frac{\omega_o^2}{4Q^2(\Delta\omega)^2} \tag{8.57}$$

Equation (8.57) forms the noise-shaping function for the oscillator. In other words, for a given noise power generated by the transistor amplifier part of the oscillator, this equation describes the output noise around the tone.

Phase noise is usually quoted as an absolute noise referenced to the carrier power, so (8.57) should be rewritten to give phase noise as

$$\text{PN} = \frac{|N_{\text{OUT}}(s)|^2}{2P_S} = \left[\frac{\omega_o}{2Q\Delta\omega}\right]^2 \left[\frac{|N_{\text{IN}}(s)|^2}{2P_S}\right] \tag{8.58}$$

where P_S is the signal power of the carrier, noting that phase noise is only half of the noise present. The other half is amplitude noise, which is of less interest. Also, in this approximation, conversion of amplitude modulation to phase modulation (AM to PM conversion) is ignored. This formula is known as Leeson's equation [14].

The one question that remains here is: What exactly is N_{IN}? If the transistor and bias are assumed to be noiseless, then the only noise present will be due to the resonator losses. Since the total resonator losses are due to its finite resistance, which has an available noise power of kT, then,

$$|N_{\text{IN}}(s)|^2 = kT \tag{8.59}$$

The transistors and the bias will add noise to this minimum. Note that since this is not a simple amplifier with a clearly defined input and output, it is not appropriate to define the transistor in terms of a simple noise figure. Considering the bias noise in the case of the $-G_m$ oscillator, as shown in Figure 8.2(c), noise will come largely from the drain current noise of the transistor in the current source when the transistors M_1 and M_2 are switched, with each being on about half the time. If ρ is the fraction of a cycle for which the transistors are completely switched, i_{nt} is the noise current injected into the oscillator from the biasing network during this time. During transitions, the transistors act like amplifiers; thus, drain current noise i_{dn} from the cross-coupled transistors usually dominates the noise during this time. The total input noise becomes

$$|N_{\text{IN}}(s)|^2 \approx kT + \frac{i_{\text{nt}}^2 R_T}{2}\rho + i_{\text{dn}}^2 R_T(1-\rho) \tag{8.60}$$

where R_T is the equivalent parallel resistance of the tank. Thus, we can define an excess noise factor for the oscillator as excess noise injected by noise sources other than the losses in the tank:

$$F = 1 + \frac{i_{\text{nt}}^2 R_T}{2kT}\rho + \frac{i_{\text{dn}}^2 R_T(1-\rho)}{kT} \tag{8.61}$$

Note that, as the Q of the tank increases, R_T increases, and noise has more gain to the output; therefore, F is increased. Thus, while (8.58) shows a decrease in phase noise with an increase in Q, this is somewhat offset by the increase in F. If noise from the bias source i_{cn}, is filtered, and if fast switching is employed, it is possible to achieve a noise factor close to unity. Now (8.58) can be rewritten as

$$\text{PN} = \left[\frac{\omega_o}{2Q\Delta\omega}\right]^2 \left(\frac{FkT}{2P_S}\right) \tag{8.62}$$

Note that in this derivation, it has been assumed that flicker noise is insignificant at the frequencies of interest. Flicker noise is generally very important, especially in CMOS designs. If ω_c represents the flicker noise corner where flicker noise and thermal noise are equal in importance, then (8.62) can be rewritten as

$$\mathrm{PN} = \left[\frac{\omega_o}{2Q\Delta\omega}\right]^2 \left(\frac{FkT}{2P_S}\right)\left(1 + \frac{\omega_c}{\Delta\omega}\right) \tag{8.63}$$

It can be noted that (8.63) predicts that noise will roll off at slopes of –30 dB/dec or –20 dB/dec, depending on whether flicker noise is important. However, in real life, at high frequency, there will be a thermal noise floor. A typical plot of phase noise versus offset frequency is shown in Figure 8.33.

So far, the phase noise discussion has been of oscillators that have no tuning scheme. However, most practical designs incorporate some method to change the frequency of the oscillator. In these oscillators, the output frequency is proportional to the voltage on a control terminal:

$$\omega_{\mathrm{osc}} = \omega_o + K_{\mathrm{VCO}} V_{\mathrm{cont}} \tag{8.64}$$

where K_{VCO} is the gain of the VCO, and V_{cont} is the voltage on the control line. If it is assumed that V_{cont} is a low-frequency sine wave of amplitude V_m, using the narrowband FM approximation, the resulting output voltage is

$$v_{\mathrm{out}}(t) \approx A\cos(\omega_o t) + \frac{AV_m K_{\mathrm{VCO}}}{2\Delta\omega}\left[\cos(\omega_o + \Delta\omega)t - \cos(\omega_o - \Delta\omega)t\right] \tag{8.65}$$

where A is the carrier power, and $\Delta\omega$ is the frequency of the controlling signal. Thus, if it is assumed that the sine wave is a noise source, then the noise power present at $\pm\Delta\omega$ is given by

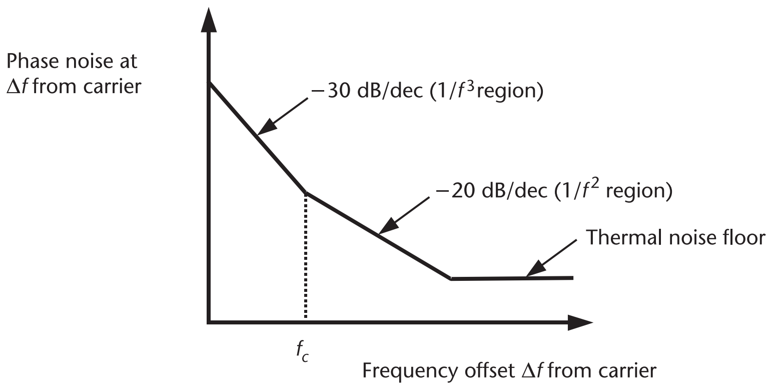


Figure 8.33 Phase noise versus frequency.

$$\text{Noise} = \left(\frac{AV_m K_{\text{VCO}}}{2\Delta\omega}\right)^2 \tag{8.66}$$

This can be converted into phase noise by dividing by the signal power:

$$\text{PN} = \left(\frac{V_m K_{\text{VCO}}}{2\Delta\omega}\right)^2 \tag{8.67}$$

We note that any low-frequency noise appearing on either of the output terminals will also appear on the other side since, at low frequency, the inductor behaves as a short circuit. Thus, it is not only noise appearing directly on the control node but also any low-frequency noise in the oscillator that can lead to low-frequency noise upconversion.

From the preceding analysis, it is easy to see how one might estimate the effect of low-frequency noise on the phase noise of the oscillator. Using a simple, small-signal noise analysis, one can find out how much noise is present at the varactor terminals. Then, knowing the K_{VCO}, the amount of phase noise can be estimated.

However, this is not necessarily the whole story. Noise on any terminal that controls the amplitude of the oscillation can lead to fluctuations in the amplitude. The varactor capacitance will depend to some extent on the envelope of the signal. Thus, these fluctuations, if they occur at low frequencies, will be converted into phase noise and can actually dominate the noise content in some cases. However, a small-signal analysis will not reveal this.

8.10.2 Switching Phase Noise in Cross-Coupled Pairs

If the cross-coupled pair is switching completely, then any low-frequency signal from the bias supply is switched back and forth between the two sides at the carrier rate. This directly converts low-frequency noise to the carrier frequency. However, since it is AM modulation, it is not directly a problem unless some further nonlinearity converts it into phase noise. Similarly, noise around the fundamental frequency is mixed by the fundamental frequency so that the resultant noise is safely far away from the fundamental. The big problem is that noise components at twice the fundamental frequency are mixed to the fundamental, producing a phase noise component. Such components, however, can be blocked largely by an LC filter in the bias path, as shown in Figure 8.15(b). There is a perception that the cross-coupled pair acts like a mixer with the result that low-frequency noise only produces AM noise (some of which can be converted into phase noise through nonlinear components), and noise injected at the fundamental gets mixed away from the fundamental frequency. Thus, some consider that there is no need to be concerned with low-frequency noise in the bias current source. However, the mixer argument is overly simplified as the mixing action is far from ideal, and there will be significant feedthrough, especially of low-frequency noise. At low frequency, the impedance of the resonator inductor is low, and any noise voltage appearing at the oscillator output appears across the varactors and, hence, causes phase noise. We note that on-chip filters have been used successfully in the bias circuit [9]. However, to extend the filter so that it removes low-frequency noise would require inductor or capacitor sizes too large to be integrated. Some attempts have been made with series off-chip degeneration inductance or parallel off-chip capacitance for this purpose [10].

8.11 Low-Frequency Phase Noise Upconversion Reduction Techniques

The following three sections discuss three techniques to reduce phase noise upconversion: bank switching, differential tuning, and simultaneous matching of transconductance and impedance.

8.11.1 Bank Switching

To cover the required frequency band and the effects of process variations, a particular K_{VCO} is required. However, if the band is broken into many subbands, then K_{VCO} is reduced, and phase noise up conversion is reduced. For example, by breaking a band into three subbands, potentially K_{VCO} is reduced by a factor of three, and phase noise up conversion can be reduced by a factor of about 10 dB. A better way to implement bank switching with n banks of varactors as shown in Figure 8.34 is to use (n − 1) AMOS varactors as switches and one bank as a continuously variable capacitor. Because of the shape of the AMOS varactor curve (as shown in Figure 8.10) when operated as a switch, AMOS varactors have a low gain when fully switched. This will result in minimal low-frequency noise

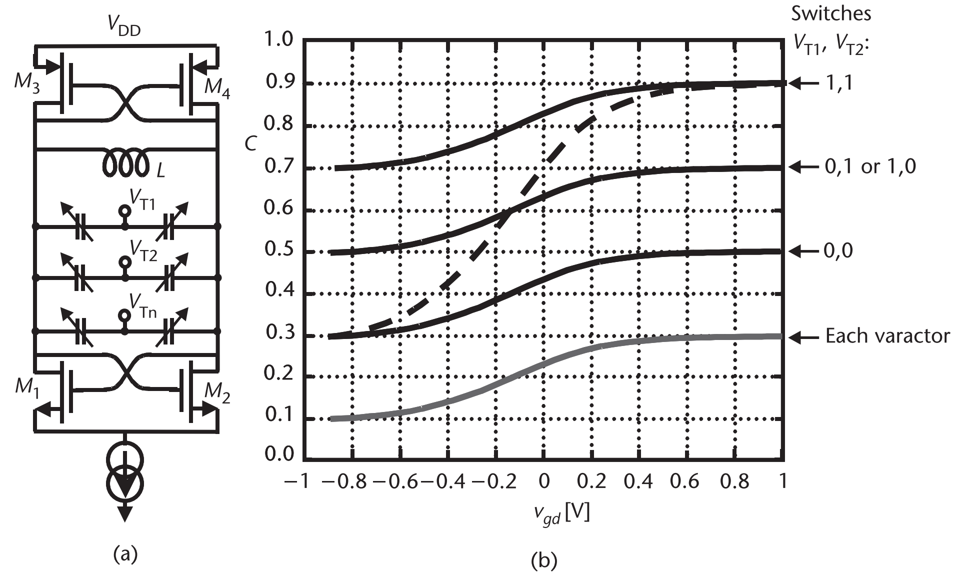


Figure 8.34 Oscillator with banks of varactors. To minimize low-frequency noise upconversion, recommended operation is to use n – 1 as switches and one for continuous tuning: (a) schematic and (b) response curves.

upconversion. Note that, to get full switching, both positive and negative control voltages must be supplied. This is conveniently obtained when the oscillator has both NMOS and PMOS cross coupling, since the output nodes are nominally biased between the power-supply voltages. However, for the case where there is only a single polarity of cross-coupling, the output voltage will be nominally at the rail voltage [e.g., with NMOS cross coupling, as in Figure 8.2(c), the outputs are nominally at V_{DD}]. In such a case, the varactors will need to be decoupled in order to obtain both positive and negative bias voltages. Such additional capacitance in series will reduce the K_{VCO}, which can be accounted for in the design and is not a problem, as lower gain was desired; however, it will limit the total tuning range.

It is possible to use weighted varactor sizes to get more curves with a smaller number of capacitors and control voltages. An example is shown in Figure 8.35 with two switchable varactors (with capacitance values of 50/150 fF and 150/450 fF). Also, there is one adjustable varactor, adjustable between 100 and 300 fF. As shown, there are four possible output curves, each varying by 100 fF, with a total range of capacitance values between 300 and 900 fF, the same range as before; however, there is now some overlap between the ranges, allowing for a more flexible oscillator design.

An automated method can be devised to tune such a circuit within a PLL-based synthesizer. A comparator-based circuit is used to observe the VCO control voltage. If the control voltage approaches one of its limits, a comparator output triggers the next curve in the appropriate direction to be selected, for example, by either counting up or counting down in a binary counter. The counter output can then select the appropriate switchable varactors. It should be noted that binary

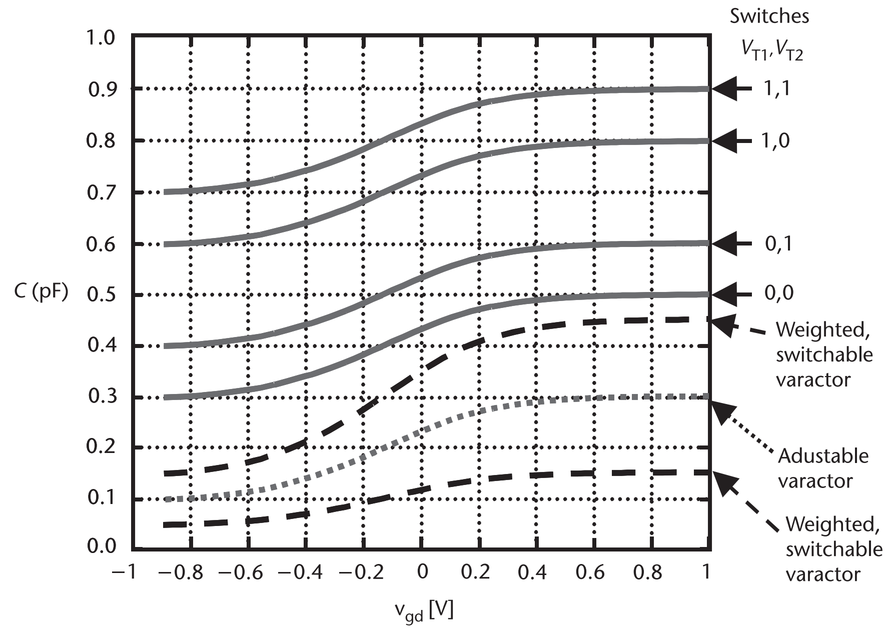


Figure 8.35 Oscillator with weighted switched varactors and adjustable varactor.

weighting is appropriate only if varactors have capacitance ratio of 2:1. In such a case, it is straightforward to get uniform curves. The example above using a 3:1 capacitance ratio illustrates the point that binary weighting is not best, as well as the difficulty in getting uniform curves.

8.11.2 g_m Matching and Waveform Symmetry

It is often stated that phase noise up conversion can be reduced by matching the transconductance of PMOS and NMOS negative g_m cells in a complementary style VCO, such as the one shown in Figure 8.2(e). This reduction occurs since any disturbance at the output will produce equal currents in the PMOS and NMOS transistor, thus minimizing the effect on the output. However, this is not really the full story as it is possibly more realistic to consider a noise current being injected into the output, for example, because of $1/f$ noise in one of the transistors. This current injected into the output node is completely canceled out if both the g_m and the impedance are matched. Typically, to match g_m requires larger PMOS transistors than NMOS transistors; hence, the gate capacitance is larger, and the impedance will be lower. However, if the process is fast enough that one can use nonminimum gate length in the NMOS transistors, then simultaneous matching is possible. For example, [15] demonstrated that simultaneous g_m and impedance matching resulted in up to 8 dB of phase noise improvement compared to matching for g_m only. Mathematically, in order to match the capacitance of both NMOS and PMOS transistors in the design requires that

$$\begin{aligned} \gamma W_n L_n C_{\text{ox}} &= \gamma W_p L_p C_{\text{ox}} \\ W_n L_n &= W_p L_p \end{aligned} \tag{8.68}$$

where γ is a transistor parameter defined in Appendix B and W_n and L_n are the width and length of the NMOS transistors, and W_p and L_p are the width and length of the PMOS transistors. If the g_m of both NMOS and PMOS are to be the same, this requires that

$$\begin{aligned} \sqrt{2\mu_n C_{\text{ox}} \frac{W_n}{L_n} I_{\text{DS}}} &= \sqrt{2\mu_p C_{\text{ox}} \frac{W_p}{L_p} I_{\text{DS}}} \\ \mu_n \frac{W_n}{L_n} &= \mu_p \frac{W_p}{L_p} \end{aligned} \tag{8.69}$$

These two conditions mean that, for best phase noise, the ratio of the lengths of the transistors should be

$$L_n = L_p \sqrt{\frac{\mu_n}{\mu_p}} \tag{8.70}$$

where L_p is usually set to the minimum allowed by the technology, and the ratio of the width of the PMOS and NMOS transistors should be

$$\frac{W_p}{W_n} = \sqrt{\frac{\mu_n}{\mu_p}} \tag{8.71}$$

8.11.3 Differential Varactors and Differential Tuning

Differential varactors controlled by a differential tuning voltage, as shown in Figure 8.36(a), can be used to reduce or eliminate low-frequency noise upconversion. This is shown in Figure 8.36(b), which shows varactor capacitance versus tuning voltage. The varactors labeled $C_{\text{var}+}$ have capacitance that increases with applied voltage, while the varactors labeled $C_{\text{var}-}$ decrease with applied voltage. Thus, if a differential voltage is applied, $C_{\text{var}+}$ diodes see a positive voltage, while $C_{\text{var}-}$ varactors see a negative voltage. Hence, both varactors are increased in capacitance for an increase in differential input voltage. However, for a common-mode voltage, both varactors see a voltage in the same direction; hence, the increase in capacitance from the $C_{\text{var}+}$ varactor is matched by an equal decrease in the capacitance from the $C_{\text{var}-}$ varactors. Low-frequency noise (e.g., $1/f$ noise injected from the cross-coupled transistors or the bias circuit) is equivalent to a common-mode input due to the low impedance of the inductor at low frequencies. With positive varactor slope given by $K_{\text{VCO}+}$ and negative varactor slope given by $K_{\text{VCO}-}$, low-frequency noise upconversion is given by

$$\text{PN} = \left[\frac{V_m(K_{\text{VCO}+} - K_{\text{VCO}-})}{2\Delta\omega}\right]^2 \tag{8.72}$$

Thus, common-mode or low-frequency noise rejection is only effective if the differential varactors are exactly symmetrical. With perfect symmetry, there is

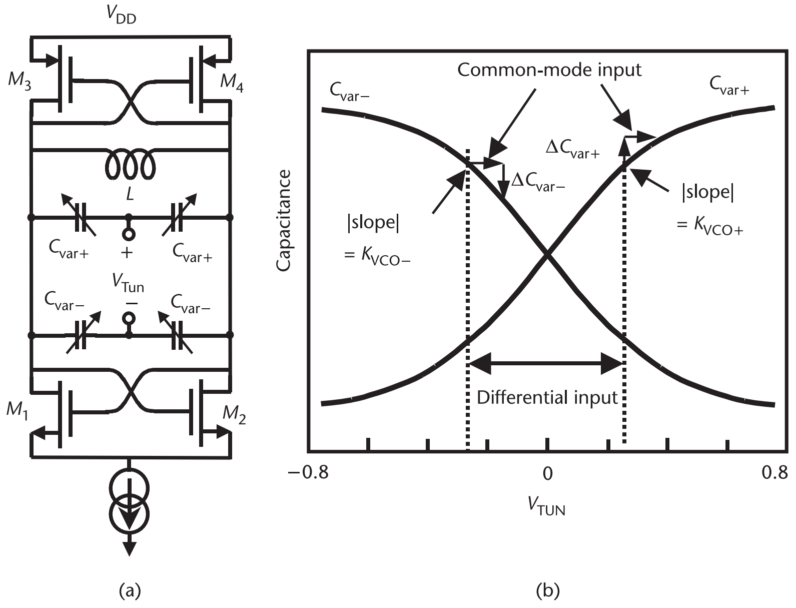


Figure 8.36 Oscillator with differential varactor tuning: (a) schematic, and (b) capacitance versus tuning voltage.

complete rejection; however, for any residual error in symmetry, rejection is reduced. To optimize symmetry, a nonzero bias voltage may be required on the varactor. For example, directly using the varactor characteristics, as shown in Figures 8.34 and 8.35, will not produce a symmetrical result since, at 0V, the capacitance is not halfway between the minimum and maximum capacitance. However, using a dc bias of –0.15V results in operation at the point of average capacitance and maximum symmetry, and this is how the curves in Figure 8.36(b) are generated. Nonsymmetry can be the result of the difference between the parasitic capacitance on the two ends of the varactor. As an example, in [7], 5-GHz oscillators using differential AMOS varactors showed a phase noise reduction of about 10 dB compared to oscillators using single-ended AMOS varactors. The varactors were in a silicon on insulator (SOI) process, which was shown to have better symmetry than bulk CMOS; hence, differential tuning in bulk CMOS would be expected to show a somewhat lower phase noise improvement.

Example 8.4: PMOS Versus Complementary VCO Design

Compare the design of PMOS only to a complementary VCO topology. Design each for best swing to achieve the lowest phase noise. The VCOs should oscillate at 5 GHz in a 0.18-μm CMOS technology. A 1-nH inductor with a Q of 10 is available for the design. Assume for this design that the mobility of the PMOS transistors is one-third that of the NMOS transistors and design the VCOs to be powered from a 1.5-V supply.

Solution: A 1-nH inductor with a Q of 10 will have 314Ω of parallel resistance. To get the VCO to oscillate at 5 GHz with a 1-nH inductor will mean that the total capacitance due to the transistors, plus additional capacitance, should be about 1 pF. Now the VCO current must be set appropriately. With the complementary design, the voltage at the resonator will swing about mid-rail to the supply and down to ground, leaving some room for the current source to operate correctly. If the current source can be operated with 300 mV of headroom, this will mean that the maximum peak-to-peak swing for this design will be 1.2V. The current should be set to give this amplitude for lowest phase noise, but more current will be wasted and lead to excess noise. Therefore, for the complementary design, the current should be set so that the peak voltage swing is 0.6V. Therefore,

$$I_{\text{comp}} = \frac{2R_P I}{\pi \cdot v_{\text{out}}\big|_{\text{Comp}}} = 3 \text{ mA}$$

In the case of the PMOS design, leaving 300 mV for the current source, the peak swing should be 1.2V. Therefore, the current should be set to

$$I_{\text{PMOS}}\, v_{\text{out}}\big|_{SE} = \frac{R_P I}{\pi \cdot v_{\text{out}}\big|_{\text{PMOS}}} = 12 \text{ mA}$$

Now with twice the swing but exactly the same resistance and Q, theoretically the phase noise of the PMOS design should be 6 dB better than the phase noise of the complementary design, but at the cost of four times the power. If the PMOS design were chosen to have the same swing as the complementary design, then it would draw 6 mA of current, thus delivering the same phase noise as the complementary design but at twice the power.

In order for these designs to be built, the transistors must be sized. For the complementary design, the PMOS transistors are chosen to have a length of 0.18 μm and the NMOS transistors to have a length of

$$L_n = L_p\sqrt{\frac{\mu_n}{\mu_p}} = 0.31\ \mu\text{m}$$

The widths of the transistors must be set large enough that 1/*f* noise is not too large and the on resistance of the devices does not act to limit the swing excessively, but not so large that the parasitic capacitance of the devices dominates the frequency of oscillation. In this design, the width of the NMOS was chosen to be 55 μm, while the PMOS width was set to be 110 μm (note that this is a ratio of 2:1, which is different from theory, in order to get both the C_{gs} and the g_m to match closely in simulation). In the case of the PMOS-only design, the transistors were made wider (175 μm) in order to accommodate the larger currents. The single-ended output voltage for each of the designs is shown in Figure 8.37. This shows that the designs have close to the predicted output swings. The PMOS design with 12 mA is a little lower due to the finite source-drain resistance of the PMOS at higher currents.

The phase noise for the three designs is shown in Figure 8.38. As predicted, the phase noises of the PMOS VCO and the complementary VCO are identical when they have the same output swing, although the PMOS VCO burns twice as much current. When the PMOS design is run at full swing, it delivers 4 to 5 dB better phase noise then the complementary design is capable of delivering. Note that this is slightly less than the 6 dB that simple theory would predict because more current means that more noise is produced in this design.

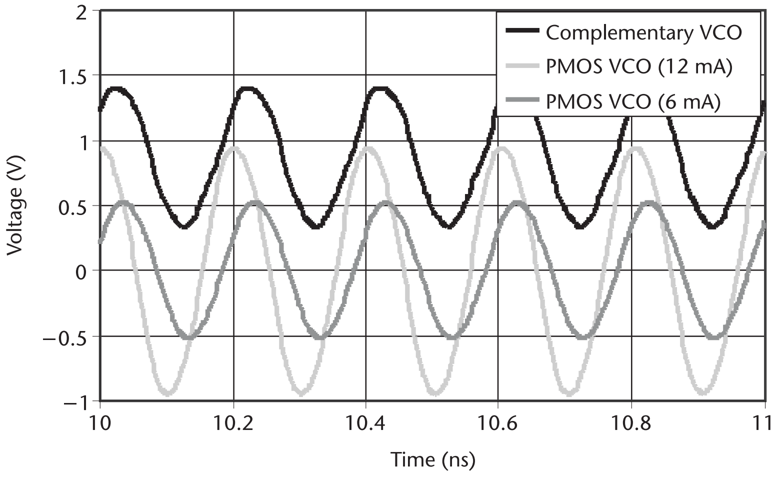


Figure 8.37 Comparison of VCO output voltage.

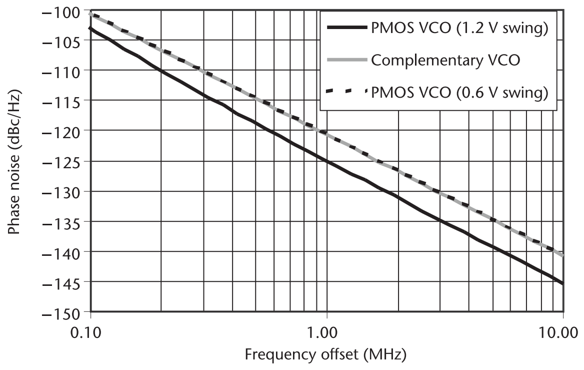


Figure 8.38 Comparison of VCO phase noise.

8.12 Ring Oscillators

Ring oscillators, like all oscillators, must satisfy the Barkhausen criteria for oscillation. However, with ring oscillators, it is usually the phase shift that we need to test for, as the gain requirement is usually quite easily satisfied. With a digital-style inverter, the gain can be very high at low frequencies. However, the voltage quickly reaches a limit, and the effective gain is then zero. Therefore, since the system does not spend its time in the linear region, the concept of gain is a bit more ambiguous than it is for LC oscillators. However, a common test for startup in ring oscillators is that, if you bias all the circuits at their switching points, the loop gain must be bigger than one at the frequency of interest.

A ring oscillator is usually made up of an odd number of inverters or delay cells with the output fed back to the input, as shown in Figure 8.39. When power is applied to this circuit, assume the input to 1 is low and the output capacitance C_1 is charged up. When the next stage input sees a high, it will discharge C_2. When the third stage sees a low, it charges C_3. This will in turn discharge stage 1.

Thus, f is related to I/C where I is the charging or discharging current, and C is the capacitance size. Thus, frequency can be controlled by changing I. This circuit can operate to high frequency if simple inverters are used. It can be made with CMOS or bipolar transistors. However, ring oscillators, not having inductors, are usually thought to be noisier than LC oscillators, but this depends on the design and the technology.

Now, an interesting question at this stage is: How many inverters will be needed to make the circuit oscillate? At first glance, many students may feel that the circuit

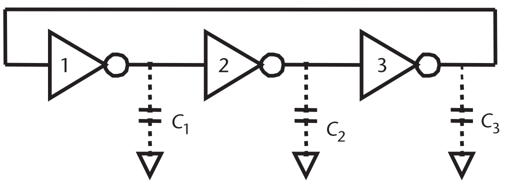


Figure 8.39 A simple ring oscillator.

shown in Figure 8.40(a) should, in fact, oscillate. If the input is zero, then the output will become one until the output then produces a zero, and so forth. So, should this be a perfectly good oscillator? Well, let's hope not, because one way to build such a circuit is shown in Figure 8.40(b), and if that circuit is unstable, then a lot of engineers need to fix a bunch of current mirrors!

So, why, in fact, does it not oscillate? The answer is quite simple: it does not have the required phase shift. The phase shift around that loop is only 180°, which is 180° short of the required 360°.

So the next question is: Can two inverters oscillate? As for any even number of inverters in a ring, this will have 360° of phase shift at low frequency, so the circuit will latch and remain in that latched position. In particular, for two stages, the two-inverter ring shown in Figure 8.41(a) can be redrawn as in Figure 8.41(b), clearly showing that it is equivalent to a latch. For this reason, even numbers of inverters in a ring are avoided.

However, two stages can be made to oscillate if one of the stages is noninverting; for example, with differential circuits, one can simply flip the polarity of the connection between two stages and still have 180° of phase shift for an even number of stages. Then, the delays of the circuit make up the additional 180° of phase shift. However, for two stages, assuming each is equivalent to the simple first-order pole shown in Figure 8.42, they each need 90° of phase shift, and this will happen only at infinite frequency. Thus, without additional phase shift, oscillation will be very unreliable with two stages. Since ring oscillators with only two stages can operate to higher frequencies (than ring oscillators with more stages), some

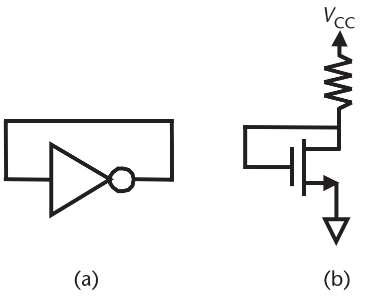


Figure 8.40 An oscillator that is not: (a) block diagram, and (b) schematic.

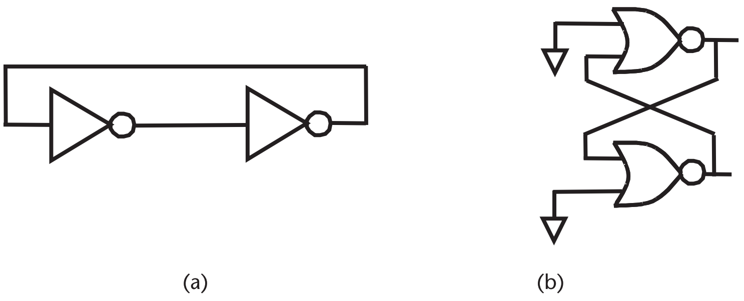


Figure 8.41 Two inverters in a ring (a) as normally drawn, and (b) circuit shown as an equivalent latch.

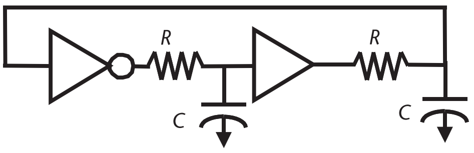


Figure 8.42 Two-stage ring oscillator with delay added at the outputs. Note that this is usually implemented in differential circuits, where one stage has the wires crossed, here represented by a noninverting stage.

effort has been put into finding techniques to add additional phase delay for reliable oscillations. Such techniques will be discussed in Section 8.14.

Therefore, in general, the minimum number of stages used to build a ring oscillator is usually three or more. Thus, a three-stage ring oscillator will be the first that we treat here in more detail. We start by modeling the ring oscillator as three negative transconductors driving RC loads, as shown in Figure 8.43. Each stage is assumed to have a current gain of $-G_m$ from input to output. Therefore, the voltage gain from input to output of one stage is

$$G(s) = \frac{v_{\text{out}}}{v_{\text{in}}} = \frac{-G_m R}{1 + sCR} \tag{8.73}$$

As a result, for three stages, the overall open-loop gain of the oscillator is

$$H(s) = \left(\frac{-G_m R}{1 + sCR}\right)^3 \tag{8.74}$$

Now each stage contributes a low-frequency phase shift of 180° because of the negative transconductance. At the frequency of interest, there is an additional phase shift per stage of

$$\phi = -\tan^{-1}(RC\omega) \tag{8.75}$$

To obtain a total phase shift that is a multiple of 360°, an additional 180° of phase shift is required from the RC networks, so the phase shift in each stage must be equal to 60°. Thus, the frequency of oscillation can be found from (8.75) to be

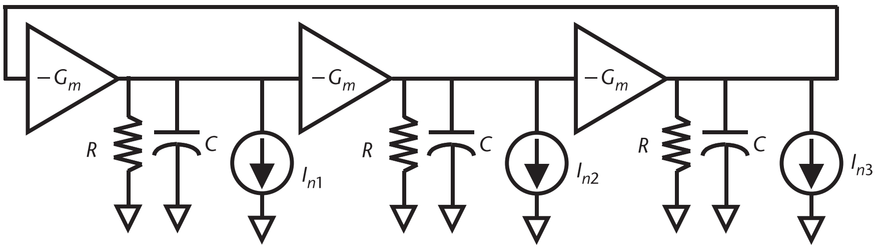


Figure 8.43 Oscillator model also used for noise analysis.

$$\omega_{\text{osc}} = \frac{\sqrt{3}}{RC} \tag{8.76}$$

Using this relationship, the above loop gain expression (8.74) can be rewritten as

$$H(j\omega) = \left(\frac{-G_m R}{1 + j\sqrt{3}\,\dfrac{\omega}{\omega_{\text{osc}}}} \right)^3 \tag{8.77}$$

For oscillations in steady state, the loop gain must equal one. Therefore,

$$\left(\frac{-G_m R}{1 + j\sqrt{3}\,\dfrac{\omega}{\omega_{\text{osc}}}} \right)^3_{\omega = \omega_{\text{osc}}} = 1 \tag{8.78}$$

With some manipulation,

$$(-G_m R)^3 = -8 \tag{8.79}$$

Therefore,

$$G_m R = 2 \tag{8.80}$$

Similarly, for a four-stage ring oscillator, the required phase shift is only 45°. Thus, in this case, the frequency is

$$\omega_{\text{osc}} = \frac{1}{RC} \tag{8.81}$$

and the condition for oscillation is

$$G_m R = \sqrt{2} \tag{8.82}$$

Using a similar analysis, any order of ring oscillator frequency and gain requirements can be found.

Alternatively, rather then using phase shift, we can characterize an inverter stage by the amount of time delay it has. Thus, for a three-stage oscillator, each stage needs a phase shift of 60°, or, equivalently, the delay in each inverter is $T/6$. Consequently, given a delay of τ, the frequency of oscillation will be $1/6\tau$. In general, with an N-stage ring oscillator, the frequency of oscillation is given by

$$f_{\text{osc}} = \frac{1}{2N\tau} \tag{8.83}$$

Single-ended ring oscillators must be designed with an odd number of stages. However, it is possible to design with an even number of stages if the unit blocks are differential. In this case, one stage is designed to be noninverting by using the opposite outputs from one of the stages. Such an oscillator is shown in Figure 8.44. Now, because there are four stages, it is a simple matter to come up with quadrature outputs, as also shown in the diagram.

Each output waveform can be assumed to be the result of a constant current I charging up a capacitor C. An estimate of frequency can be made by noting that the output voltage will swing by about half the power-supply voltage before triggering the next delay cell. With N stages and power-supply voltage V_{DD}, the frequency can be predicted as follows:

$$T = 2N\frac{C}{I}\frac{V_{DD}}{2} = \frac{NCV_{DD}}{I} \tag{8.84}$$

or

$$f_{osc} = \frac{1}{T} = \frac{I}{NCV_{DD}} \tag{8.85}$$

As a quick example, if a capacitance as low as 20 fF can be used, to achieve 2 GHz requires a current of about $I = f_{osc} \cdot N \cdot C \cdot V_{DD} = 2\text{ GHz} \cdot 4 \cdot 20\text{ fF} \cdot 1.2\text{V} \approx 200\ \mu\text{A}$.

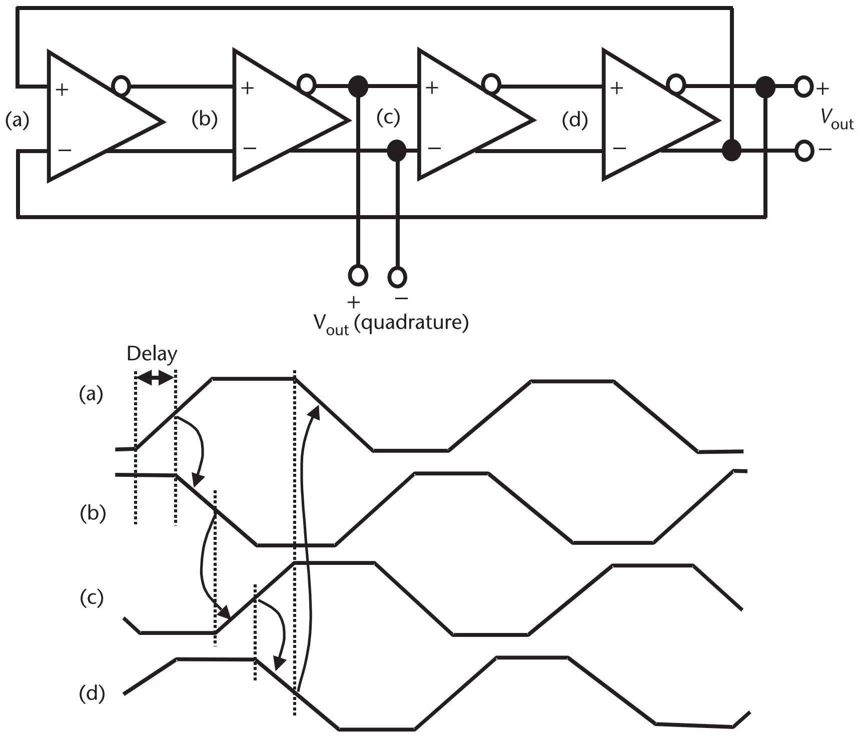


Figure 8.44 A differential ring oscillator with quadrature outputs, and waveforms at each node.

8.13 Common Inverter Circuits

Each of the amplifiers or inverters can be made very simply or in a more complex way. Some examples of simple inverters are shown in Figure 8.45. However, these have no means of tuning. Simple tuning circuits are shown in Figure 8.46.

Since oscillating frequency depends on how quickly the interstage capacitance is being charged, bias current can be used to control the oscillating frequency. For the differential inverter, it may be tempting simply to combine two of the single-ended inverters, for example, as shown in Figure 8.47. However, this circuit as shown has no connection between the two sides, so there is no reason to believe that the two output signals would be a differential signal. A simple simulation will demonstrate that a four-stage ring oscillator built with this circuit will not work, since the four-stage oscillator depends on the cross coupling of one stage to provide the correct number of inverting elements in the loop. With the cross coupling, since this delay cell does not behave in a differential fashion, this is equivalent to eight single-ended inverters in a ring; since eight is an even number, this circuit will latch up. A second simulation with three inverters in a ring demonstrates that this circuit does work; however, it turns out that, generally, the "+" and "–" rings, being

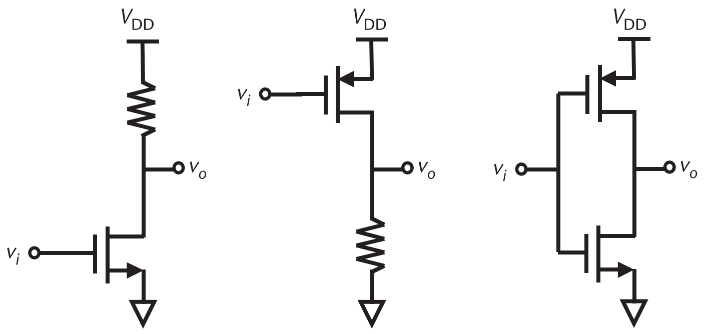


Figure 8.45 Simple, single-ended, delay cell implementations.

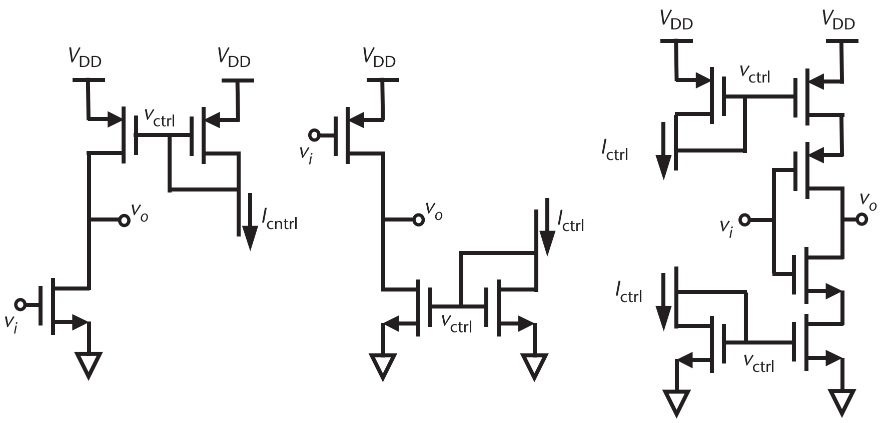


Figure 8.46 Simple, tunable, single-ended, delay cell implementation.

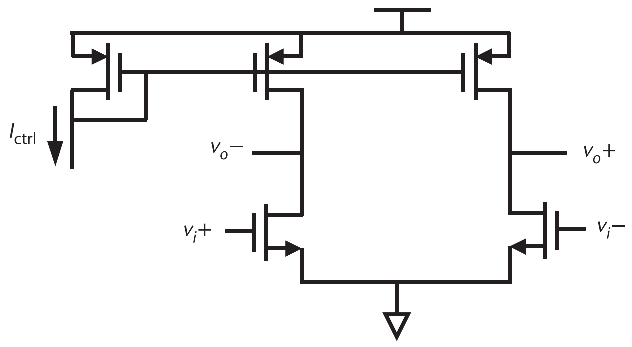


Figure 8.47 A poor differential delay cell implementation.

effectively separate rings, will oscillate in phase. Thus, the differential output signal is zero, as can be demonstrated by connecting the outputs to a differential amplifier.

Instead, one needs a way to ensure that the two output signals are truly differential, for example, with a differential pair. An example of this is shown in Figure 8.48.

The PMOS transistors and bias voltage can be arranged in several different ways. In the first way, transistors behave like a triode region resistor where the resistance is kept large enough to allow a reasonable output swing. Such a circuit is shown in Figure 8.49(a) in which the PMOS gate is connected to ground. The second way is for the bias voltage and transistor to be designed to form a high impedance current source, an example of which is shown in Figure 8.49(b). In such a circuit, the current source I_{SS} must be designed to be larger so that it can pull down the output voltage when appropriate. For example, I_{SS} can be made to be twice the current I_3 or I_4 in the PMOS transistors. In Figure 8.49(b), if I_{ctrl1} is equal to I_{ctrl2}, this implies that the NMOS mirror ratio is twice that of the PMOS mirror ratio.

Other variations include using additional diode-connected transistors in parallel with the PMOS load transistors, which in [16] is called a symmetrical load. This

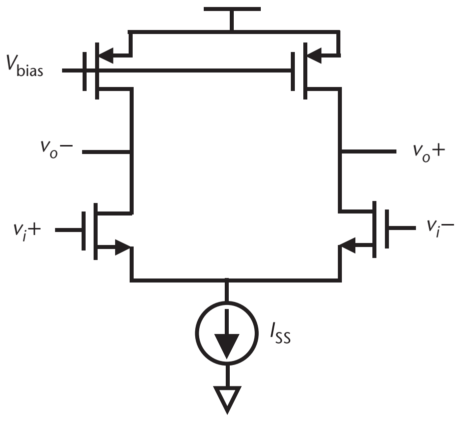


Figure 8.48 Delay cell based on a differential pair.

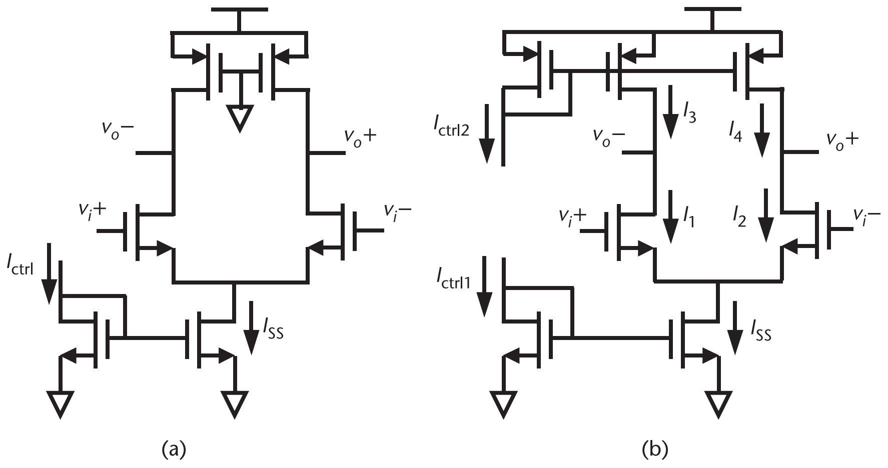


Figure 8.49 Delay cell based on a differential pair, including biasing details: (a) PMOS gate tied to the ground; and (b) current source load.

load arrangement, shown in Figure 8.50, results in a nearly linear transfer function of current versus control voltage. This circuit arrangement was also used by [17] in comparison with other circuits.

Two circuits with cross-coupled positive feedback are shown in Figure 8.51. The circuit shown in Figure 8.51(a) [18, 19] is a saturated-gain stage with regenerative cross-coupled PMOS transistors that can be controlled to tune the delay (and, hence, the frequency). This circuit provides for rail-to-rail output signals and full switching of the transistors in the stage. The feedback properties of the latching transistors M_1 and M_2 speed up the signal transitions at the output. The stage also avoids the use of cascode connections and a tail-current-source transistor that would limit the signal swing and add more noise to the output. Larger signal swing and faster transitions help to improve signal-to-noise ratio (SNR). The circuit in Figure 8.51(b) represents another style of delay cells with cross-coupled transistors providing positive feedback. Variations of this circuit include combining the two

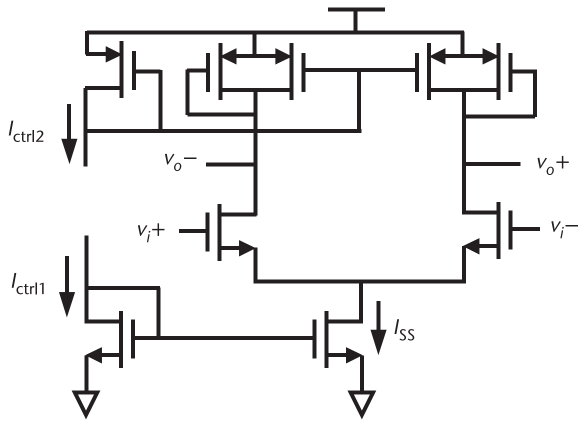


Figure 8.50 Delay cell based on a differential pair with a symmetrical load.

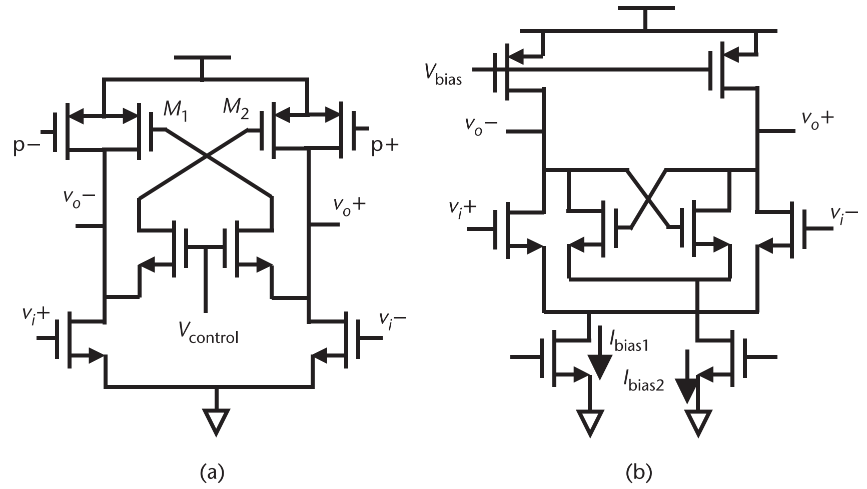


Figure 8.51 Delay cell based on differential pair with cross-coupling (positive feedback) to adjust delay: (a) with control voltage and (b) with bias current.

bias circuits, adding coarse and fine control of bias currents [20, 21], realization with bipolar delay cells, and adding active inductive loads [22].

Recently, there have been many designs in which the PMOS current sources are controlled not from the previous stage but by an earlier stage (being careful to get the polarity right) in order to compensate for the slower speed of PMOS compared to NMOS. While such circuits can be single ended [23], a differential version is shown in Figure 8.52 [19] using a delay cell similar to that of Figure 8.51(a).

We note that an oscillator using the circuit in Figure 8.51(a) was compared to an LC oscillator in [24]. The results demonstrated that the LC design was superior in nearly every way: power dissipation was less than one-tenth (3.6 mW compared to 50 mW), phase noise was better by 6 dB when scaled to the same frequency, and tuning range was surprisingly much better at 46% versus 13.3%. The main advantage for this ring oscillator is that it may have lower layout area since it does not require an inductor. Note that, typically, ring oscillators have a wider tuning range than LC oscillators since LC oscillators are inherently limited by the tuning range of varactors, and ring oscillators have no equivalent fundamental limitation.

8.14 Method for Designing a Two-Stage Ring Oscillator

In [25], a two-stage ring oscillator design was used to try to get high speed in a 0.4-μm process. It was determined that anything more than two stages would not be fast enough for the desired 2.5 Gbps. However, a two-stage ring oscillator as seen in Figure 8.53 typically does not have enough phase shift to oscillate reliably. So, the authors added an RC circuit in such a way that it behaved like an inductor, which added the appropriate phase shift to guarantee oscillation. Their modified circuit is shown in Figure 8.54.

It can be shown that the C_L, R_1, C_1 network at the output produces a phase-shifted signal. This signal is fed back by the transistor, and the resulting equivalent

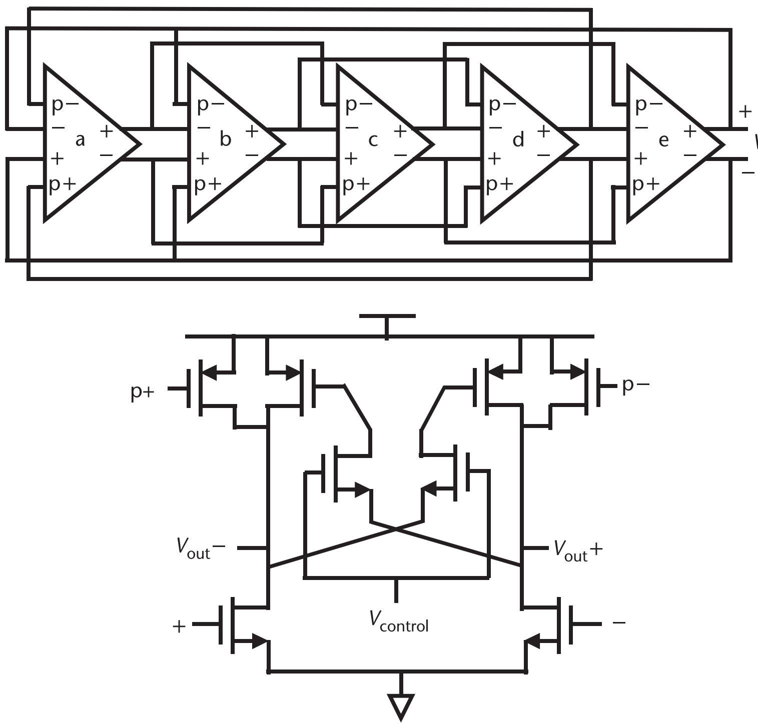


Figure 8.52 (a) Five-stage, multiple-pass ring oscillator and (b) saturated gain stage with cross-coupled PMOS transistors.

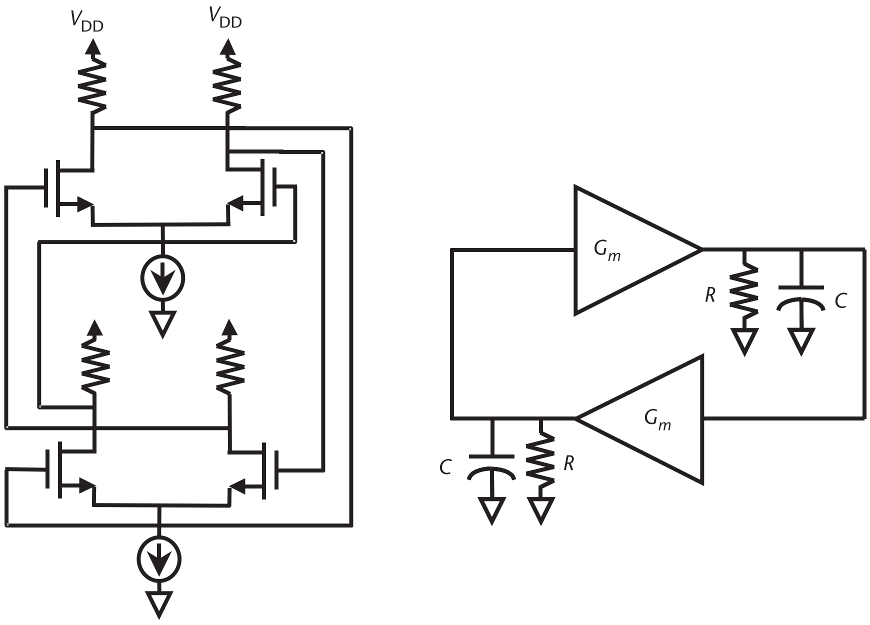


Figure 8.53 Two-stage oscillator that will have trouble oscillating: (a) schematic and (b) equivalent circuit.

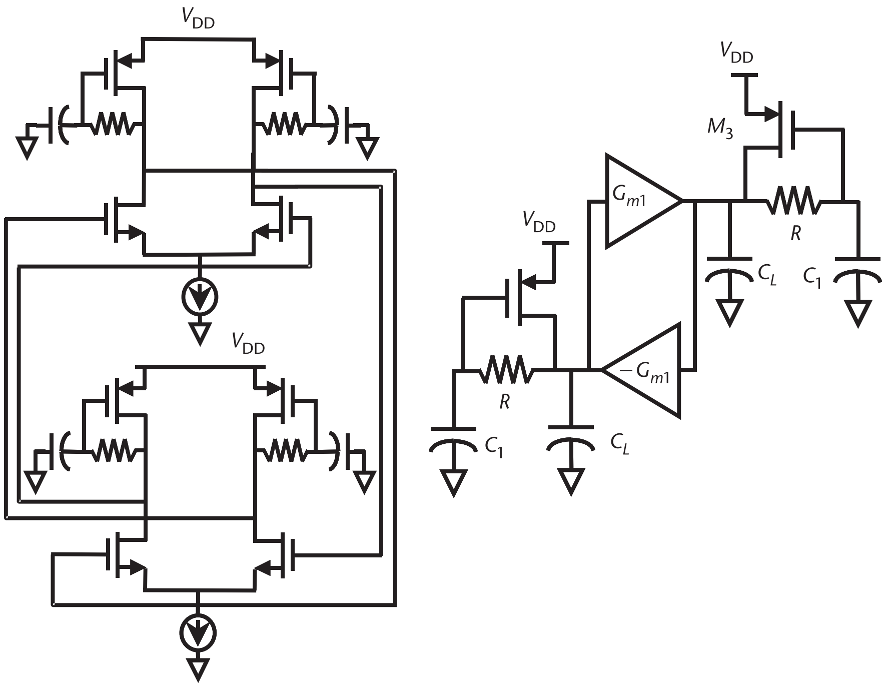


Figure 8.54 Two-stage oscillator with loading that is effectively inductive: (a) schematic and (b) equivalent circuit.

impedance, under the right conditions, can have an inductive component. If the output resistance of the extra transistor, for example r_{o3} of M_3, is much larger than R_1 and is also much larger than $1/g_{m3}$, the single-ended equivalent of each gain stage has gain equal to

$$A_1 = \frac{-g_{m1}(sC_1R_1 + 1)}{s^2C_1C_LR_1 + s(C_1 + C_L) + g_{m3}} \tag{8.86}$$

This has two poles and a zero. If the poles are complex conjugates and at a lower frequency than the zeros, then this stage can produce more than 90° of phase shift, as is required to form an oscillator. It can be shown that these conditions are met if

$$C_1 < C_L \tag{8.87}$$

and

$$g_{m3}R_1 > \frac{C_1}{4C_L} + \frac{1}{2} + \frac{C_L}{4C_1} \tag{8.88}$$

As a quick example, if $C_L = 2C_1$, then $g_{m3}R_1 > 9/8$. Assuming that $g_{m3}R_1 = 2$, then the poles and zeros can be determined to be at

$$Z = -\frac{1}{C_1R_1} \quad P_{1,2} = -\frac{3}{4C_1R_1}(1 \pm j0.778) \tag{8.89}$$

Thus, as expected, the complex conjugate pair of poles happens before the zero and will add extra phase lag to the system, ensuring that there is more than 90° of phase shift at some finite frequency.

8.15 Phase Noise and Jitter in Ring Oscillators

The topic of phase noise in ring oscillators has been addressed in a number of ways. In the following paragraphs, it will be analyzed using the same technique as used for LC oscillators, that is, to find the effective loop gain and then to calculate the effect of noise introduced into such a loop [26]. The assumption of steady-state gain parameters works quite well for ring oscillators with few stages as the output tends to be quasisinusoidal. For rings with a large number of delay cells, if there is more complete switching, such analyses may be less accurate. More detailed discussion of phase noise and alternative techniques can be found in [27–31]. Starting with the loop gain of a three-stage ring oscillator (as in Figure 8.43) in steady state, in (8.76), the frequency of oscillation is shown to be

$$\omega_{\text{osc}} = \frac{\sqrt{3}}{RC} \tag{8.90}$$

and in (8.77) the loop gain is determined to be

$$H(j\omega) = \left(\frac{-G_mR}{1 + j\sqrt{3}\dfrac{\omega}{\omega_{\text{osc}}}} \right)^3 \tag{8.91}$$

A further condition in (8.80) is that G_mR is equal to two. Now, if we differentiate $H(j\omega)$ in (8.91) with respect to ω, we get

$$\frac{dH}{d\omega} = -3(-G_mR)^3 \left(\frac{1}{1 + j\sqrt{3}\dfrac{\omega}{\omega_{\text{osc}}}} \right)^4 \left(j\sqrt{3}\frac{1}{\omega_{\text{osc}}} \right) \tag{8.92}$$

Now we take the magnitude of this expression at ω_{osc}:

$$\left|\frac{dH}{d\omega}\right|^2 \Bigg|_{\omega = \omega_{\text{osc}}} = \frac{27(G_mR)^6}{256\omega_{\text{osc}}^2} = \frac{27}{4\omega_{\text{osc}}^2} \tag{8.93}$$

By combining (8.51) and (8.93), we get

$$\left|\frac{N_{\text{out}}}{N_{\text{in}}}\right|^2 = \frac{1}{(\Delta\omega)^2\left|\frac{dH}{d\omega}\right|^2} = \frac{4\omega_{\text{osc}}^2}{27(\Delta\omega)^2} \tag{8.94}$$

Now, this is input noise in terms of voltage. Since we have a current,

$$N_{\text{in}} = i_n\left(\frac{R}{1 + j\omega RC}\right) \tag{8.95}$$

At the frequency of oscillation,

$$|N_{\text{in}}|^2 = \frac{i_n^2 R^2}{4} \tag{8.96}$$

Therefore,

$$\left|\frac{N_{\text{out}}}{i_n}\right|^2 = \frac{R^2}{27}\left(\frac{\omega_{\text{osc}}}{\Delta\omega}\right)^2 \tag{8.97}$$

With three stages each contributing equal noise,

$$\left|\frac{N_{\text{out}}}{i_n}\right|^2 = \frac{R^2}{9}\left(\frac{\omega_{\text{osc}}}{\Delta\omega}\right)^2 \tag{8.98}$$

Now we approximate the noise coming out of the active circuitry as the drain noise from two CMOS transistors, each with output drain current noise given by

$$(i_{\text{Dn}})^2 = 4\gamma kTg_m \tag{8.99}$$

For long channel devices, the factor γ is approximately two-thirds, but for short channel devices, it can be higher, closer to unity. If the two transistors have approximately equal g_m and, by noting that $G_m R = 2$ and assuming that small-signal transconductance and effective transconductance are the same ($g_m = G_m$), we come up with the following approximation:

$$i_n^2 \approx 8kT/R \tag{8.100}$$

Thus, the total output noise is

$$|N_{\text{out}}|^2 = 8kT\frac{R}{9}\left(\frac{\omega_{\text{osc}}}{\Delta\omega}\right)^2 \tag{8.101}$$

Therefore, the phase noise will be equal to

$$\mathrm{PN}(\Delta\omega) = \frac{8kT}{v_{\mathrm{osc}}^2}\frac{R}{9}\left(\frac{\omega_{\mathrm{osc}}}{\Delta\omega}\right)^2 \tag{8.102}$$

where v_{osc}^2 is the amplitude of oscillation squared. This is an approximate expression for a three-stage ring oscillator. For a four-stage ring oscillator,

$$\left|\frac{dH}{d\omega}\right|^2\Bigg|_{\omega=\omega_{\mathrm{osc}}} = \frac{8}{\omega_{\mathrm{osc}}^2} \tag{8.103}$$

and

$$\left|\frac{N_{\mathrm{out}}}{i_n}\right|^2 = \frac{R^2}{4}\left(\frac{\omega_{\mathrm{osc}}}{\Delta\omega}\right)^2 \tag{8.104}$$

for all four stages. Then, with the same assumptions about the noise current per stage as given by (8.100), the phase noise is given by

$$\mathrm{PN}(\Delta\omega) = \frac{2kTR}{v_{\mathrm{osc}}^2}\left(\frac{\omega_{\mathrm{osc}}}{\Delta\omega}\right)^2 \tag{8.105}$$

Comparing (8.105) to (8.102), the phase noise for four stages compared to three stages is 2.25 times, or 3.5 dB, higher if resistance R is the same. It should be noted that with more stages, the small-signal approximation is less valid, and the waveform has a transition portion that could be seen as small-signal or linear; however, each output also spends some time at a nearly constant voltage close to the power-supply rails. During the linear portion, the gain is higher than that predicted by $G_m R = 2$, while during the limited portion, the gain is approximately zero. Thus, the linear model, which uses a constant effective gain, is not totally valid. Similarly, the input noise is also cyclostationary. Thus, to get an accurate picture of phase noise, a more complex model must be used; however, the above guidelines give us an idea of the major terms affecting phase noise. To use this equation to predict phase noise, we find R by using (8.90), where capacitance and frequency of operation are known. For the three-stage ring oscillator, (8.102) can be rewritten as follows:

$$\mathrm{PN}(\Delta\omega) = \frac{8kT}{v_{\mathrm{osc}}^2}\frac{\sqrt{3}}{9\omega_{\mathrm{osc}}C}\left(\frac{\omega_{\mathrm{osc}}}{\Delta\omega}\right)^2 \tag{8.106}$$

Thus, the only variable is the capacitance, and given the capacitance, the phase noise can be predicted. So, why are transistor size and bias current not in this equation? We note that they are, indirectly, as they need to be adjusted to give the desired operating frequency. However, once that is done, and if the linear approximations are valid, only the capacitance should be important. A similar equation for the four-stage ring oscillator can be obtained by combining (8.105) with the knowledge that $\omega_{osc} = 1/RC$, resulting in:

$$\mathrm{PN}(\Delta\omega) = \frac{2kT}{v_{osc}^2 \omega_{osc} C} \left(\frac{\omega_{osc}}{\Delta\omega}\right)^2 \tag{8.107}$$

Thus, comparing (8.106) with (8.107), we see that with resistance R removed from the equations, phase noise is about 1.3 times, or about 1.1 dB, higher for a four-stage ring oscillator compared to a three-stage ring oscillator.

In summary, it would appear that any desired phase noise can be achieved simply by choosing an appropriate capacitor size. However, for a larger capacitor, (8.85) shows that more bias current is required to achieve the desired frequency. Thus, there is a direct trade-off between bias current (and power dissipation) and phase noise. In comparison with LC oscillators, as mentioned at the end of Section 8.13, at the same power dissipation, phase noise is typically worse by 10 to 20 dB. Similarly, to achieve similar phase noise as an LC oscillator, reported ring oscillators have required up to 10 times more power dissipation [24].

Example 8.5: Ring Oscillator Design

Design a single-ended ring oscillator operating at 1 GHz with phase noise of –100 dBc/Hz at a 1-MHz offset.

Solution: In a 0.13-μm process, the power supply is typically 1.2V. We assume that the output voltage is 1V peak to peak, or 0.35 V_{rms}. Then, for three stages, phase noise can be predicted using (8.106):

$$\begin{aligned}\mathrm{PN}(\Delta\omega) &= \frac{8 \times 4 \times 10^{-21} \times \sqrt{3} \times 2 \cdot \pi \cdot 1 \times 10^9}{0.35^2 \times 9 \cdot C \cdot (2 \cdot \pi \cdot 1 \times 10^6)^2} \\ &= 1.27 \times 10^{-33} \frac{\omega_{osc}}{C} \\ &= \frac{8.00 \times 10^{-24}}{C}\end{aligned}$$

A phase noise of –100 dBc is 1×10^{-10}W. Solving for capacitance we find,

$$C = \frac{8.00 \times 10^{-24}}{1 \times 10^{-10}} = 8.00 \times 10^{-14} = 80 \text{ fF}$$

Thus, for a little bit of safety, we select a 100-fF capacitor, which results in a predicted phase noise of –101 dBc/Hz at a 1-MHz offset. This result will not be achieved unless special attention is paid to other details, like the minimization of $1/f$ noise. Otherwise, errors can be 10 dB or more, as this example will illustrate. Using 100 fF, we can estimate the current it will take to achieve 1 GHz using (8.85):

$$I = fNV_{DD}C = 1 \times 10^{9} \times 3 \times 1.2 \times 100 \times 10^{-15}\ 360\ \mu\text{A}$$

Transistor sizing is done by considering current for maximum f_T. In this process, it is not practical to operate at maximum f_T since this requires a gate-to-source voltage of more than 1.2V. Instead, if the transistor size is increased by about five times, current density is down to about one-fifth of the current density for optimal f_T. More importantly, this results in the reduction of V_{GS} to about half of V_{DD}, but f_T is only reduced by about 20% of its maximum value, so this seems like a reasonable starting point. During the simulation iterations, using the circuit of Figure 8.46(a), the above current is adjusted to 330 μA with a total transistor width of 3 μm and minimum channel length. Simulation results show a phase noise of –88.7 dBc/Hz at a 1-MHz offset. Further exploration of phase noise versus capacitance, as shown in Figure 8.55, demonstrates that, as expected from (8.106), phase noise is inversely proportional to capacitance and proportional to frequency of oscillation.

However, phase noise is considerably higher than predicted by the simple theory by about 11 dB. A printout of dominant noise sources shows that 1/*f* noise is the main cause of the added phase noise. In an attempt to improve this, transistor sizes (both width and length) are doubled, resulting in a reduction in phase noise to –94.7 dBc/Hz at a 1-MHz offset, an improvement of about 6 dB. A further doubling of both *W* and *L* results in phase noise of –100.8 dBc/Hz at a 1-MHz offset, a further improvement of about 6 dB. Because of the increased parasitic capacitance, it is also necessary to increase the current, in this final case to 470 μA to keep the

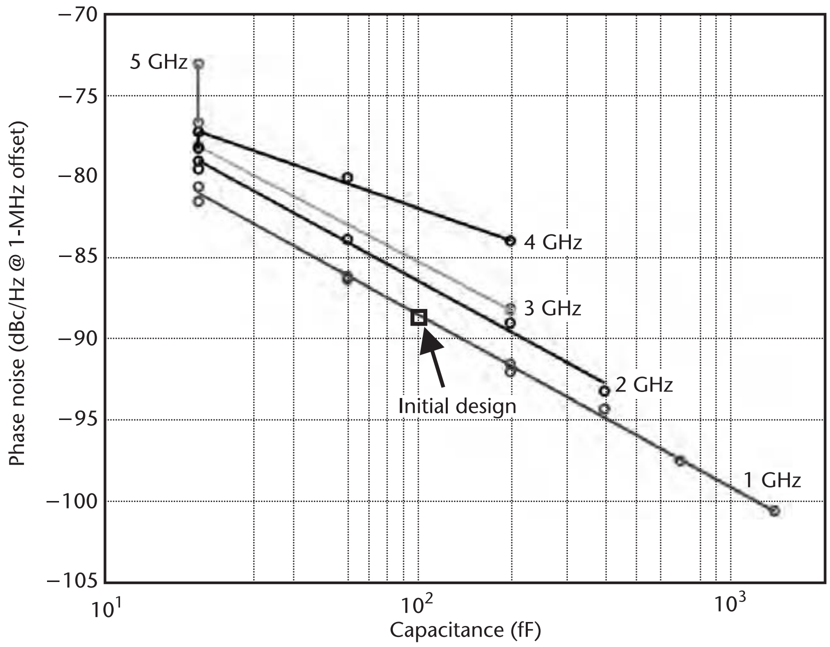


Figure 8.55 Phase noise versus load capacitance for various frequencies. Transistor lengths are minimum, so 1/*f* noise is significant at a 1-MHz offset. Phase noise is further improved by about 12 dB by increasing both *W* and *L*.

frequency at about 1 GHz. A simulation of noise versus offset is shown in Figure 8.56. It is of importance to note that the 1/*f* noise corner is at about 700 kHz. This indicates that, at a 1-MHz offset, there is still a little bit of room for improvement.

Example 8.6: Quadrature Oscillator Design

Design a quadrature oscillator oscillating at 2 GHz. Phase noise should be better than –100 dBc/Hz at a 1-MHz offset. Explore the quality of the quadrature phase matching with an output load of 2 kΩ and a capacitor mismatch on one of the stages.

Solution: The technique is similar to the previous example, except that a four-stage differential ring oscillator as in Figure 8.44 is used. The circuit of Figure 8.49(b) is used as a delay cell. As in the previous example, to meet phase noise requirements, (8.107) is used, resulting in a load capacitance of 100 fF. Since the frequency is twice that of the previous example, current is expected to be approximately double as well. This will require large transistors, which will add to the parasitic capacitance, requiring a further increase of current. As a first simulation, use transistor W/L of 12 μm/0.24 μm, double the size used in the previous example, to handle larger current. Biasing at 1 mA of current results in a phase noise of –98.8 dBc/Hz at a 1-MHz offset; however, the frequency is only 1.58 GHz. Thus, current is increased to 1.5 mA, transistor size is increased to a W/L of 16 μm/0.24 μm, the resulting phase noise is –98.6 dBc/Hz, and frequency is at 2.07 GHz. A phase noise summary indicates that the 1/*f* noise of the PMOS transistors is dominant, so these are increased in size to a W/L of 24 μm/0.36 μm. The resulting increase in parasitic capacitance is offset by an increase of current to 1.6 mA. The resulting performance is a frequency of 2.02 GHz and a phase noise of –99.99 dBc/Hz at a 1-MHz offset. The 1/*f* corner

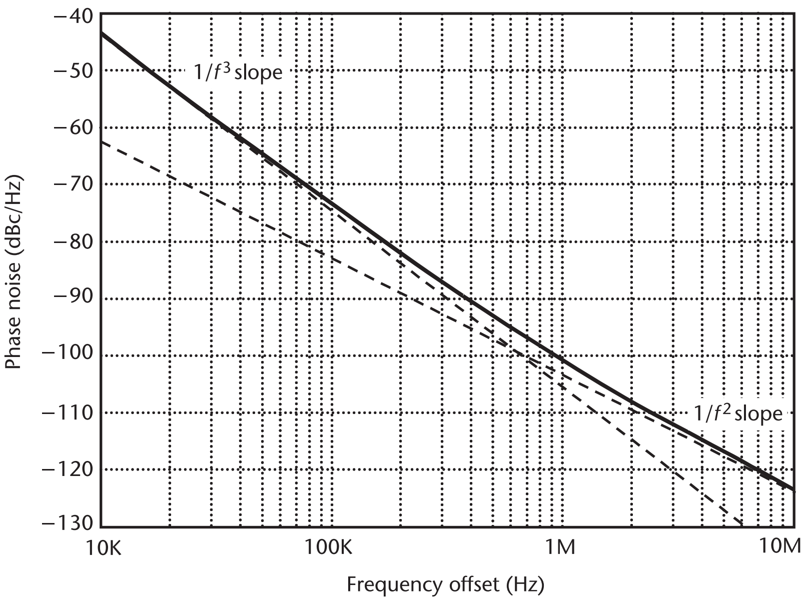


Figure 8.56 Phase noise versus offset frequency showing high 1/*f* corner frequency.

is at about 2.5 to 3 MHz, so further improvements are still possible. The output waveforms are shown in Figure 8.57.

With mismatched loads, phase errors will result. During layout, mismatch can be the result of mismatched connecting lines and lack of symmetry. After fabrication, further mismatch can occur due to process variation across the design. During simulation, the effect of mismatch can be explored by loading a stage differentially and to ground with capacitors or resistors. With a capacitive load on one stage, for example, representing a connection to additional transistors in dividers or mixers, the voltage on this stage will tend to change more slowly, following the capacitor equation $i = C\Delta v/\Delta t$. So, if $2t$ is the rise time and $2\delta t$ is the change of rise time due to a change of capacitance δC, as shown in Figure 8.58 with i and Δv kept the same, the period changes from $8t$ to $8t + 2\delta t$, and the time between

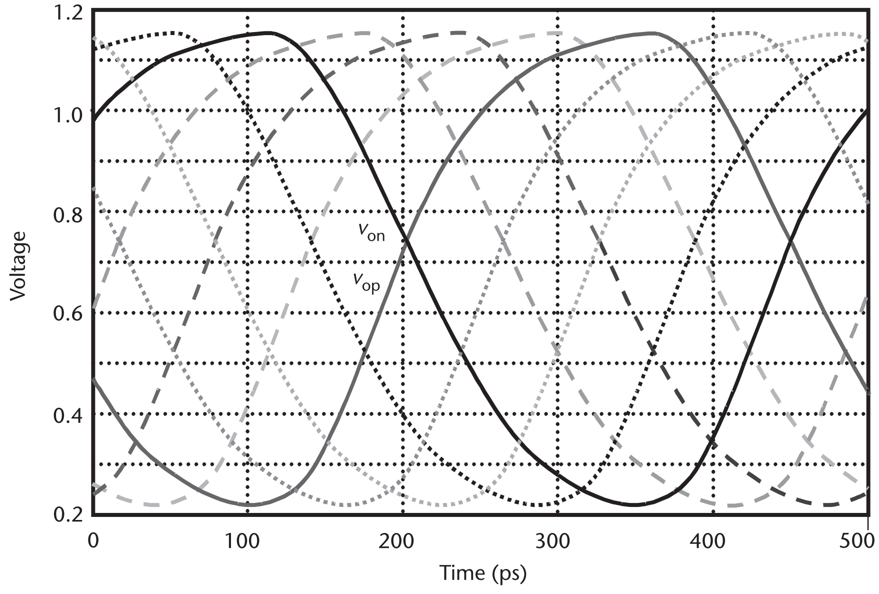


Figure 8.57 Time domain waveforms of four-stage differential ring oscillator.

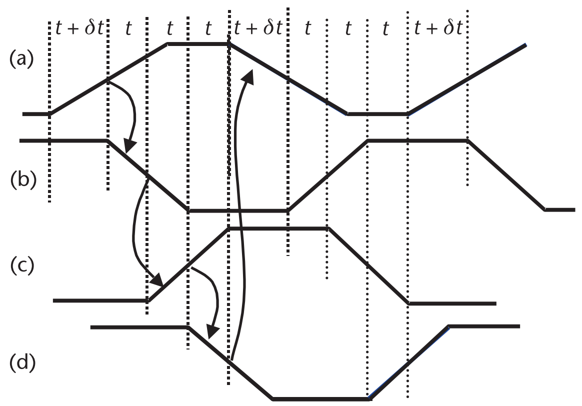


Figure 8.58 Waveforms with larger capacitor on one stage [stage (a)].

the desired outputs remains at $2t$. Thus, the new phase is $2t/(8t + 2\delta t)$ multiplied by 360°, or approximately $90°(1 - \delta t/4t)$. Thus, a 10% increase of rise and fall time in one stage will result in a 2.5% change of phase, or about 2°. Simulations were done with differential and single-ended capacitive and resistive loads with results as shown in Table 8.1. These numbers are close to the predicted numbers but all a bit lower.

8.16 Crystal Oscillators

Quartz crystal resonators are widely used in frequency-control applications because of their unequaled combination of high Q, stability, and small size. When a potential difference is applied across opposite faces of a quartz crystal, mechanical deformation takes place. If the frequency of the potential is appropriate, the crystal will vibrate and, indeed, resonate. The resonant frequency, Q, and temperature coefficient depend on the physical size and orientation of faces relative to the crystal axis.

The resonators are classified according to "cut," which is the orientation of the quartz wafer with respect to the crystallographic axes of the material. Examples are AT-, BT-, CT-, DT-, and SC-cut, but they can also be specified by orientation, for example a +5° X-cut. Although a large number of different cuts have been developed, some are used only at low frequencies, others are used in applications other than frequency control and selection, and still others have been made obsolete by later developments. At frequencies above approximately 1 MHz, primarily AT- and SC-cuts are used.

For most applications, the two-terminal equivalent circuit consisting of the static capacitance C_0, in parallel with the dynamic or motional branch, L_1-C_1-R_1, is used as shown in Figure 8.59, in which f_s is called *motional resonance frequency* given by

Table 8.1 Effect of Device Mismatch on the Phase Error of a Quadrature Oscillator (Single-Ended Is Labeled "s-e;" Differential Is Labeled "diff")

	Capacitive Load				*Resistive Load*			
Value	10 fF	50 fF	10 fF	50 fF	10 kΩ	2 kΩ	10 kΩ	7 kΩ
s-e or diff	s-e	s-e	diff	diff	diff	diff	s-e	s-e
Phase error	1.1°–1.5°	5°–7°	1.7°	7°	0°	4.4°	0.7°	1°–1.2°

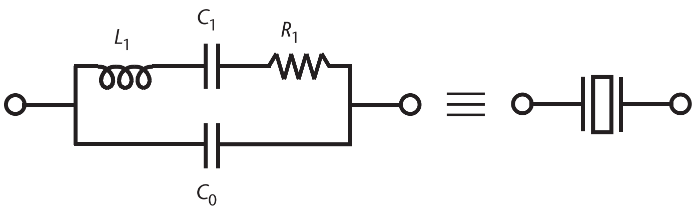


Figure 8.59 Two-terminal equivalent circuit of a crystal.

$$f_s = \frac{1}{2\pi\sqrt{L_1 C_1}} \tag{8.108}$$

For some applications, harmonics, or overtones, are used, in which case the model has more branches in parallel, one for each harmonic.

For oscillator applications, the figure of merit, M, is a useful indicator that is defined as

$$M = \frac{1}{2\pi f_s C_0 R_1} \tag{8.109}$$

For $M < 2$, the crystal reactance is never inductive at any frequency, and an additional inductor would be required to form an oscillator. In general, a larger M results in a more useful resonator.

In a crystal resonator, the quality factor of a reactive component is the reactance X_1 of the motional inductance or capacitance, divided by the motional resistance R_1:

$$Q = \frac{|X_1|}{R_1} = \frac{2\pi f_s L_1}{R_1} = \frac{1}{2\pi f_s C_1 R_1} \tag{8.110}$$

where the time constant $\tau = C_1 R_1$ depends on the mode of vibration and on the angles of cut. For AT-cut c-mode, $\tau = 10$ fs, for SC-cut c-mode, $\tau = 9.9$ fs, and for BT-cut b-mode, $\tau = 4.9$ fs [32]. Practically, quartz crystal resonators can have an unloaded Q up to a few hundred thousand and a temperature drift of less than 0.001% over the expected temperature range. The maximum Q that can be obtained is determined by several additive loss factors, the first of which is the intrinsic Q of quartz, which is approximately $16 \cdot 10^6$, divided by the frequency in megahertz for the AT-cut, and slightly higher for the SC-cut. Other factors that further limit Q are mounting loss, atmospheric loading (for nonevacuated crystal units), and the surface finish of the blank. Mounting loss depends upon the degree of trapping produced by the electrode and the plate diameter. The highest Q is obtained by using mechanically or chemically polished blanks with an adequately large diameter and an evacuated enclosure.

In the vicinity of an isolated mode of vibration, the impedance of a crystal resonator is a pure resistance at two frequencies. The lower of these is the *resonance frequency* f_r (close to, but not exactly equal to, the series self-resonance frequency f_s due to the presence of C_0); the greater is the *antiresonance frequency* f_a. In the lossless case, the frequency of antiresonance is equal to the *parallel resonance frequency* f_p, approximately equal to

$$f_p = f_s\left(1 + \frac{C_1}{2C_0}\right) = f_s\left(1 + \frac{1}{2M}\right) \tag{8.111}$$

For resonators with a large figure of merit ($M > 5$), f_r can be approximated by

$$f_r = f_s\left(1 + \frac{1}{2QM}\right) \tag{8.112}$$

Table 8.2 presents some typical parameters as found on product data sheets, for example in [33].

The impedance of the crystal can be plotted as in Figure 8.60. It can be seen that the crystal is inductive in the region between ω_r (very close to ω_s) and ω_p, and this will be a very narrow frequency range. If the crystal is used to replace an inductor in an oscillator circuit, for example, as shown in Figure 8.61(a), then oscillations will only occur in the frequency range where the crystal is actually inductive. While the crystal behaves like an inductor at the oscillating frequency, unlike a real inductor, no dc current flows through the crystal because, at dc, it is like a capacitor. Figure 8.61(b) can be derived by assuming that the positive input in Figure 8.61(a) is grounded. This is the familiar Pearce amplifier, a subset of the Colpitts oscillator and is a very common way to construct a crystal oscillator. Figure 8.62 shows two ways to realize this Colpitts-based crystal oscillator, one with a bipolar transistor, the second with MOS transistors.

As to the phase noise, besides the thermal noise with its floor around –160 dBc/Hz, 1/*f* noise exists in crystal oscillators. The total noise PSD of a crystal oscillator can be determined with Leeson's formula [14, 34]:

$$\mathrm{PN} = \frac{|N_{\mathrm{OUT}}(s)|^2}{2P_S} = \left[\frac{|H_1|\omega_o}{2Q\Delta\omega}\right]^2\left[\frac{|N_{\mathrm{IN}}(s)|^2}{2P_S}\right] \tag{8.113}$$

Table 8.2 Typical Specifications and Parameters for Precision SC-Cut Crystal Resonators

Parameter	*Specs*	*Comments*
Frequency	5–160 MHz	Harmonic mode for higher frequencies
Recommended load capacitance	About 20 pF	Typically, series capacitor for higher frequencies
Frequency adjustment tolerance	1.5–8 ppm	Generally, higher for higher frequency
Q	80k to 2.5M	Higher Q for lower frequency
R_1	35Ω–120Ω	
C_1 (fF)	0.13–0.5 fF	
C_0 (pF)	3.2–4.7 pF	

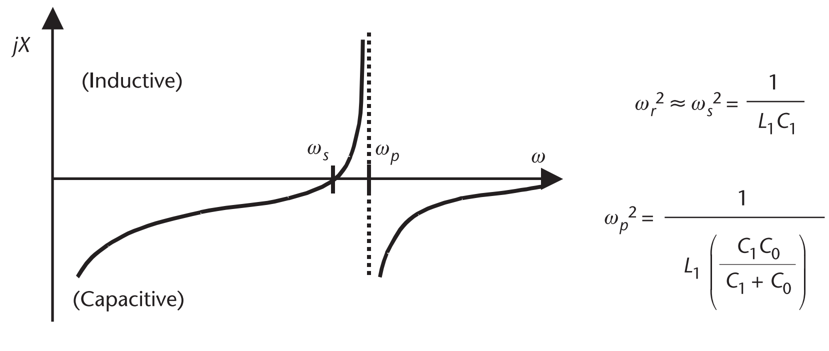


Figure 8.60 Impedance of crystal circuit model.

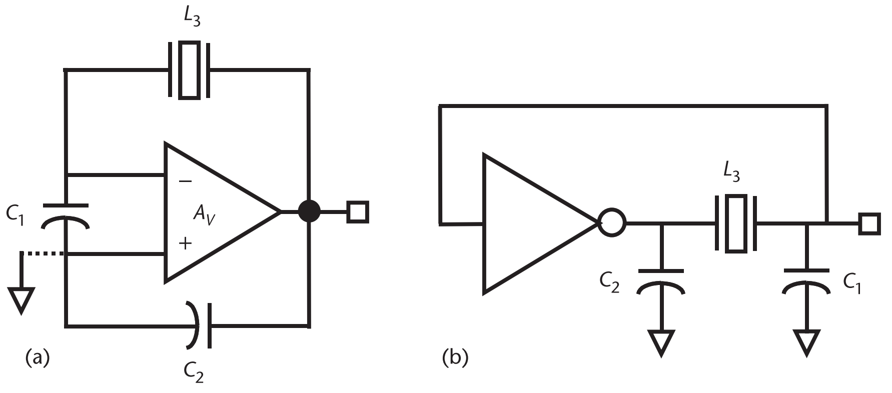


Figure 8.61 Crystal oscillator diagrams (a) with a general amplifier and (b) with an inverter.

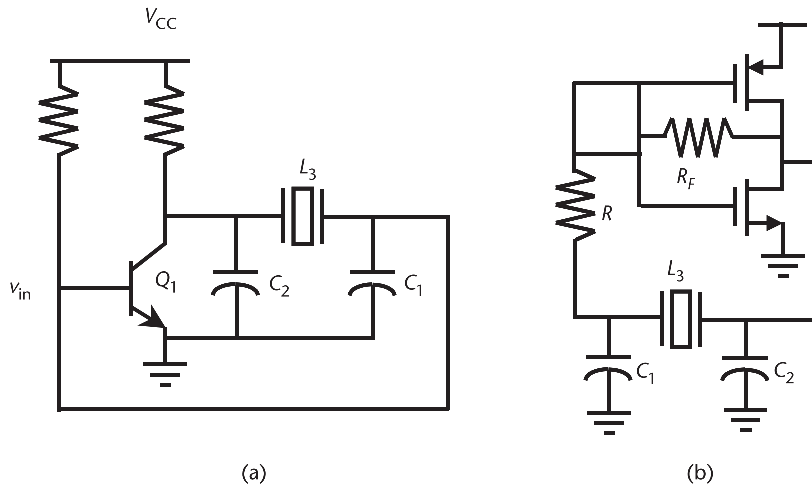


Figure 8.62 Two implementations of crystal oscillators: (a) with bipolar transistor and (b) with CMOS converter.

An empirical formula to describe crystal oscillator phase noise is given by

$$\mathrm{PN}(\Delta f) = 10^{-16\pm1} \cdot \left[1 + \left(\frac{f_0}{2\Delta f \cdot Q_L}\right)^2\right]\left(1 + \frac{f_c}{|\Delta f|}\right) \tag{8.114}$$

where f_0 is the oscillator output frequency, Δf is the offset frequency, and f_c is the corner frequency between $1/f$ and thermal noise regions, which is normally in the range 1 to 10 kHz. Q_L is the loaded Q of the resonator. Since the Q for crystal resonators is very large, as shown in Table 8.2, the reference noise contributes only to the very close-in noise, and it quickly reaches the thermal noise floor at offset frequency around f_c. Figure 8.63 demonstrates an example of a phase noise spectral density of a crystal reference source. This includes a plot of (8.114) using a first term of 10^{-16}, an f_o of 10 MHz, Q_L of 120k, and f_c of 1 kHz, showing

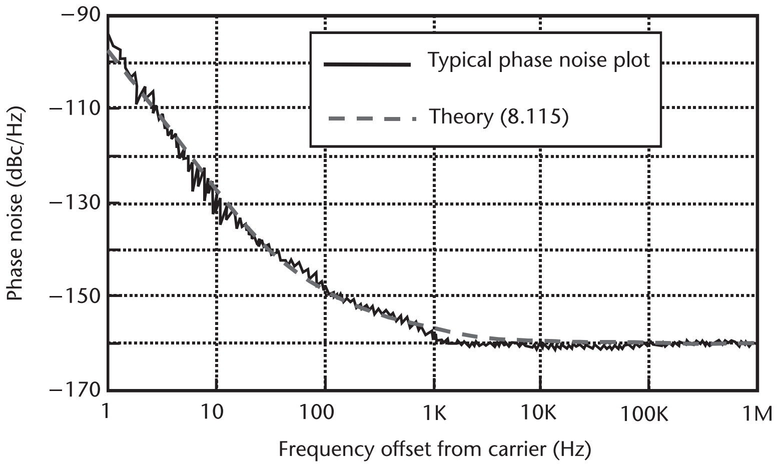


Figure 8.63 Phase noise of a crystal reference source. Theory with f_o = 10 MHz, Q_L = 120k, f_c = 1 kHz.

good agreement. Note that Q_L, the loaded Q, can be significantly lower than the unloaded Q, due to loading of the active circuitry.

In summary, because of their effective high Q (up to hundreds of thousands) and relatively low frequency (less than a few hundred megahertz), crystal oscillators will have significantly lower phase noise and lower power dissipation than LC or ring oscillators. However, typically, the purpose of a crystal oscillator is to achieve ultrahigh stability, of the order of parts per million. To accomplish this, a complete commercial crystal oscillator will also have means to do temperature compensation and amplitude control, and this is where a lot of the design effort will be directed.

8.17 Summary: Comparison of Oscillator Performance

Table 8.3 provides a quick summary comparison of the LC oscillator, the ring oscillator, and the crystal oscillator. So which oscillator is best? From the table, it can be seen that there are significant differences between the oscillators, and each

Table 8.3 Comparison Summary for Crystal, LC, and Ring Oscillators

Parameter	*Crystal*	*LC*	*Ring*
Output frequency	Low	High	Medium
Q	High	Medium	Low
Phase noise	Best	Good	Poor
Power consumption	Low	High	Highest
Multiphase output	No	No*	Yes
Frequency stability	Best	Good	Poor
Tuning range	Narrow	Medium	Wide
Integratability	No	Large size	Small size
Applications	Reference source	GHz VCO	Multiphase VCO, digital clock generation

*A quadrature LC oscillator is, in a way, a combination of an LC and a ring oscillator.

can be said to be the best in certain applications. Thus, it is important to choose the most appropriate oscillator for the job at hand.

References

[1] Long, J. R., and M. A. Copeland, "The Modeling, Characterization, and Design of Monolithic Inductors for Silicon RF IC's," *IEEE J. Solid-State Circuits*, Vol. 32, March 1997, pp. 357–369.

[2] Danesh, M., et al., "A *Q*-Factor Enhancement Technique for MMIC Inductors," *Proc. RFIC Symposium*, Baltimore, MD, June 1998, pp. 183–186.

[3] Yue, C. P., and S. S. Wong, "On-Chip Spiral Inductors with Patterned Ground Shields for Si-Based RF IC's," *IEEE J. Solid-State Circuits*, Vol. 33, May 1998, pp. 743–752.

[4] Niknejad, A. M., and R. G. Meyer, "Analysis, Design, and Optimization of Spiral Inductors and Transformers for Si RF IC's," *IEEE J. Solid-State Circuits*, Vol. 33, October 1998, pp. 1470–1481.

[5] Andreani, P., and S. Mattison, "On the Use of MOS Varactors in RF VCOs," *IEEE J. Solid-State Circuits*, Vol. 35, No. 6, June 2002, pp. 905–910.

[6] Fong, N., et al., "Accumulation MOS Varactors for 4 to 40 GHz VCOs in SOI CMOS," *Proc. 2002 International SOI Conference*, Williamsburg, VA, October 2002, pp. 158–160.

[7] Fong, N., et al., "A 1V 3.8–5.7 GHz Differentially Tuned VCO in SOI CMOS," *Proc. 2002 RFIC Symp.*, Seattle, WA, June 2002, pp. 75–78.

[8] Rogers, J. W. M., and C. Plett, *Radio Frequency Integrated Circuit Design*, Norwood, MA: Artech House, 2003.

[9] Hegazi, E., H. Sjoland, and A. A. Abidi, "A Filtering Technique to Lower LC Oscillator Phase Noise," *IEEE J. Solid-State Circuits*, Vol. 36, No. 12, December 2001, pp. 1921–1930.

[10] Andreani, P., and H. Sjoland, "Tail Current Noise Suppression in RF CMOS VCOs," *IEEE J. Solid-State Circuits*, Vol. 376, No. 3, March 2002, pp. 342–348.

[11] Andreani, P., et al., "Analysis and Design of a 1.8 GHz CMOS LC Quadrature VCO," *IEEE J. Solid-State Circuits*, Vol. 37, No. 12, December 2002, pp. 1737–1747.

[12] Tang, J., et al., "Analysis and Design of an Optimally Coupled 5-GHz Quadrature LC Oscillator," *IEEE J. Solid-State Circuits*, Vol. 37, No. 5, May 2002, pp. 657–661.

[13] Gierkink, S. L., et al., "A Low-Phase-Noise 5-GHz CMOS Quadrature VCO Using Superharmonic Coupling," *IEEE J. Solid-State Circuits*, Vol. 37, No. 5, May 2002, pp. 1148–1154.

[14] Leeson, D. B., "A Simple Model of Feedback Oscillator Noise Spectrum," *Proc. IEEE*, February 1966, pp. 329–330.

[15] Fong, N., et al., "Phase Noise Improvement of Deep Submicron Low-Voltage VCO," *Proc. ESSCIRC* 2002, Florence, Italy, September 2002, pp. 811–814.

[16] Maneatis, J. G., "Low-Jitter Process-Independent DLL and PLL Based on Self-Biased Techniques," *IEEE J. Solid-State Circuits*, Vol. 31, No. 11, November 1996, pp. 1723–1732.

[17] Dai, L., and R. Harjani, "Design of Low-Phase-Noise CMOS Ring Oscillators," *IEEE Tran. Circuits and Systems II*, Vol. 49, No. 5, May 2002, pp. 328–338.

[18] Park, C. H., and B. Kim, "A Low-Noise, 900-MHz VCO in 0.6-m CMOS," *IEEE J. Solid-State Circuits*, Vol. 34, No. 5, May 1999, pp. 586–591.

[19] Eken, Y. A., and J. P. Uyemura, "A 5.9-GHz Voltage-Controlled Ring Oscillator in 0.18-μm CMOS," *IEEE J. Solid-State Circuits*, Vol. 39, No. 1, January 2004, pp. 230–233.

[20] Song, S. J., S. M. Park, and H. J. Yoo, "A 4-Gb/s CMOS Clock and Data Recovery Circuit Using 1/8-Rate Clock Technique," *IEEE J. Solid-State Circuits*, Vol. 38, No. 7, July 2003, pp. 1213–1219.

[21] Shu, Z., K. L. Lee, and B. H. Leung, "A 2.4-GHz Ring-Oscillator-Based CMOS Frequency Synthesizer with a Fractional Divider Dual-PLL Architecture," *IEEE J. Solid-State Circuits*, Vol. 39, No. 3, March 2004, pp. 452–462.

[22] Tang, J., D. Kasperkovitz, and A. Roermund, "A 9.8–11.5-GHz Quadrature Ring Oscillator for Optical Receivers, *IEEE J. Solid-State Circuits*, Vol. 37, No. 3, March 2002, pp. 438–442.

[23] Lee, S. J., B. Kim, and K. Lee, "A Novel High-Speed Ring Oscillator for Multiphase Clock Generation Using Negative Skewed Delay Scheme," *IEEE J. Solid-State Circuits*, Vol. 32, No. 2, February 1997, pp. 289–291.

[24] Eken, Y. A., and J. P. Uyenmura, "Multiple-GHz Ring and LC VCOs in 0.18 μm CMOS," *Proc. RFIC*, Fort Worth, TX, June 2004, pp. 475–478.

[25] Anand, S. B., and B. Razavi, "A CMOS Clock Recovery Circuit for 2.5-Gb/s NRZ Data," *IEEE J. Solid-State Circuits*, Vol. 36, No. 3, March 2001, pp. 432–439.

[26] Razavi, B., "A Study of Phase Noise in CMOS Oscillators," *IEEE J. Solid-State Circuits*, Vol. 31, No. 3, March 1996, pp. 331–342.

[27] Lee, T. H., and A. Hajimiri, "Oscillator Phase Noise: A Tutorial," *IEEE J. Solid-State Circuits*, Vol. 35, No. 3, March 2000, pp. 326–336.

[28] Hajimiri, A., S. Limotyrakis, and T. H. Lee, "Jitter and Phase Noise in Ring Oscillators," *IEEE J. Solid-State Circuits*, Vol. 34, No. 3, June 1999, pp. 790–804.

[29] McNeill, J. A., "Jitter in Ring Oscillators," *IEEE J. Solid-State Circuits*, Vol. 32, No. 6, June 1997, pp. 870–879.

[30] Gierkink, S. L. J., et al., "Intrinsic 1 Device Noise Reduction and Its Effect on Phase Noise in CMOS Ring Oscillators," *IEEE J. Solid-State Circuits*, Vol. 34, No. 7, July 1999, pp. 1022–1025.

[31] Thamsirianunt, M., and T. A. Kwasniewski, "CMOS VCO's for PLL Frequency Synthesis in GHz Digital Mobile Radio Communications," *IEEE J. Solid-State Circuits*, Vol. 32, No. 10, October 1997, pp. 1511–1524.

[32] Vig, J. R., "Quartz Crystal Resonators and Oscillators," U.S. Army Electronics Technology and Devices Report, SLCET-TR-88-1, 1988.

[33] "Resonator Products," Piezo Technology Orlando, Florida, available at http://www.piezotech.com/Resonators/resonatorsindex.htm (accessed October 2004).

[34] Watanabe, Y., et al., "Phase Noise Measurements in Dual-Mode SC-Cut Crystal Oscillators," *IEEE Trans. on Ultrasonics, Ferroelectrics and Frequency Control*, Vol. 47, No. 2, March 2000, pp. 374–378.

CHAPTER 9

ΣΔ Modulation for Fractional-*N* Synthesis

9.1 Introduction

The concept of ΣΔ modulation was first employed in oversampling analog-to digital (A/D) and digital-to-analog (D/A) converters [1, 2]. Oversampling data converters operate at a much higher sampling rate than the Nyquist rate, which is twice the signal bandwidth. In a ΣΔ A/D converter, the ΣΔ modulator modulates the analog input into a digital code, either a single-bit or multibit word. Advanced semiconductor technology is better suited for providing fast digital circuits than for providing precise analog circuits. Taking advantage of ever-increasing IC speed, ΣΔ techniques trade resolution in time for resolution in amplitude and, thus, allow imprecise analog circuits to be used. ΣΔ techniques have also found applications in frequency synthesis, which is the main topic of this chapter. In a fractional-*N* frequency synthesizer, a ΣΔ modulator can be used to control the loop divider such that the fractional spurs can be randomized and shifted to a higher frequency band where they can be easily removed by the loop LPF. In this chapter, the basic concept of ΣΔ modulation is first introduced. Then, various ΣΔ modulator topologies for frequency-synthesis applications are presented, and their performances are compared.

9.2 Basic Concepts

This section provides an overview of the basic concepts necessary to understand ΣΔ modulators. First, oversampling techniques are described, followed by a description, in general terms, of how a feedback loop can result in noise shaping in a sampled system. In further sections, the concepts of oversampling and noise shaping will be applied to ΣΔ modulators.

9.2.1 Quantization Noise and Oversampling Effects

Suppose that a quantizer converts a continuous analog signal to a discrete digital signal with a characteristic shown in Figure 9.1, where x is the analog input, and y is the quantized digital output. The output is a function of the input, but it has discrete levels at equally spaced intervals Δ. Thus, unless the input happens to be an integer multiple of the quantizer resolution (step size) Δ, there will be an error

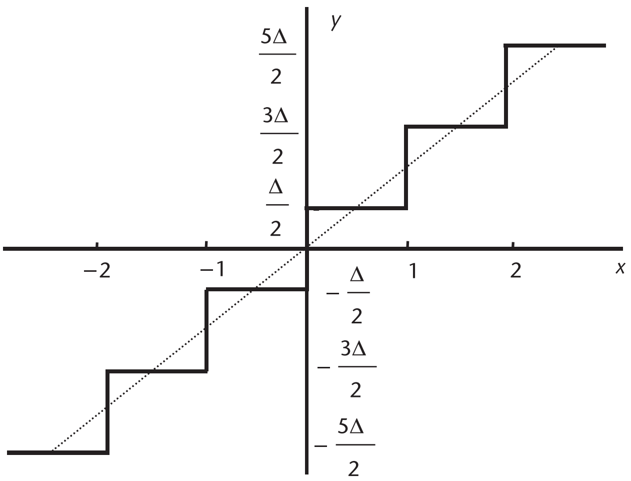


Figure 9.1 Transfer characteristic of a multibit quantizer.

in representing the input. This error e will be bounded over one quantizer level by a value of

$$-\frac{\Delta}{2} \leq e \leq \frac{\Delta}{2} \tag{9.1}$$

Thus, the quantized signal y can be represented by a linear function with an error e as

$$y = \Delta \cdot x + e \tag{9.2}$$

where the step size Δ corresponds to the slope of the straight line shown in Figure 9.1. The quantization error as a function of the input is given in Figure 9.2. Note that the error is a sawtooth waveform with a slope of $-\Delta$. If the input is "random" in nature, then the instantaneous error will also be random. The error is thus uncorrelated from sample to sample and can hence be treated as "noise." Quantization and the resultant quantization noise can be modeled as a linear circuit including an additive quantization error source, as shown in Figure 9.3.

Thus, the quantization noise for a random signal can be treated as additive white noise having a value anywhere in the range from $-\Delta/2$ to $\Delta/2$. The quantization noise has equal probability with a probability density of

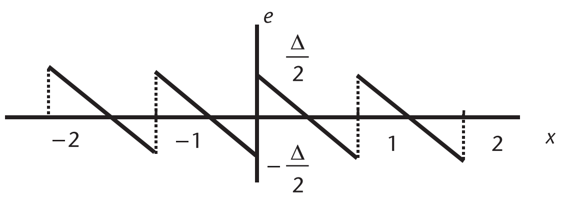


Figure 9.2 The quantization error as a function of the input.

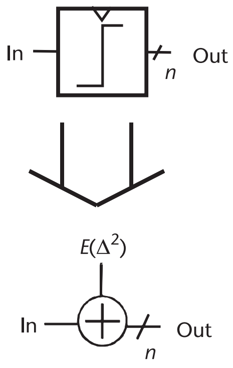


Figure 9.3 Modeling of the quantization as a linear circuit with an additive noise.

$$p(e) = \begin{cases} \frac{1}{\Delta} & \text{if } -\frac{\Delta}{2} \leq e \leq \frac{\Delta}{2} \\ 0 & \text{otherwise} \end{cases} \tag{9.3}$$

where the normalization factor $1/\Delta$ is needed to guarantee that

$$\int_{-\Delta/2}^{\Delta/2} p(e)\, de = 1 \tag{9.4}$$

The mean square rms error voltage e_{rms} can be found by integrating the square of the error voltage and dividing by the quantization step size:

$$e_{\text{rms}}^2 = \int_{-\infty}^{+\infty} p(e) e^2\, de = \frac{1}{\Delta} \int_{-\Delta/2}^{\Delta/2} e^2\, de = \frac{\Delta^2}{12} \tag{9.5}$$

From control theory (see Appendix A), it is known that the frequency spectrum of a sampled system repeats once every sampling frequency. Thus, the spectrum of the quantization noise in a sampled system will be centered around dc and spread out to half of the sampling frequency $f_s/2$, and there will be a copy of the noise spectrum from $f_s/2$ to $3f_s/2$, and so on. Considering that all the noise power lies in the range of the positive frequency band (i.e., $0 \leq f \leq \infty$), the quantization noise power thus folds into the band from dc to $f_s/2$. Assuming white noise, the PSD $E^2(f)$ of the quantization noise is given by

$$E^2(f) = \frac{e_{\text{rms}}^2}{f_s/2} = 2Te_{\text{rms}}^2 \tag{9.6}$$

where the sample period $T = 1/f_s$. For a band limited signal $0 \leq f < f_0$ with bandwidth of f_0, the quantization noise power that falls into the signal band can thus be found as

$$n_0^2 = \int_0^{f_0} E^2(f)\, df = 2f_0 Te_{\text{rms}}^2 = \frac{\Delta^2 f_0}{6 \cdot f_s} = \frac{\Delta^2}{12 \cdot \text{OSR}} \tag{9.7}$$

where the *oversampling ratio* (OSR) is defined as the ratio of the sampling frequency f_s to the Nyquist frequency $2f_0$; that is

$$\text{OSR} = \frac{f_s}{2f_0} \tag{9.8}$$

In an N-bit sampled system, if the quantizer has 2^N quantization levels equally spaced by Δ, then the maximum peak-to-peak amplitude is given by

$$v_{\max} = (2^N - 1) \cdot \Delta \tag{9.9}$$

If the signal is sinusoidal, the associated signal power is

$$P = \frac{1}{8}(2^N - 1)^2 \cdot \Delta^2 \tag{9.10}$$

Thus, the SNR due to quantization noise power that falls into the signal band becomes

$$\text{SNR} = 10 \log\left(\frac{\frac{1}{8}(2^N - 1)^2 \Delta^2}{n_0^2}\right) \approx 10 \log\left(\frac{3 \cdot 2^{2N}\, \text{OSR}}{2}\right) \tag{9.11}$$

Noting that $\log_{10}(x) = \log_{10}(2) \cdot \log_2(x)$, the above expression becomes

$$\text{SNR} \approx 6.02 \cdot N + 3 \cdot \log_2(\text{OSR}) + 1.76 \tag{9.12}$$

Therefore, the SNR improves by 6 dB for every bit added to the quantizer. For the same amount of total quantization noise power, every doubling of the sampling frequency reduces the in-band quantization noise by 3 dB. As illustrated in Figure 9.4, oversampling reduces the in-band rms quantization noise since the total noise is spread across the entire sampling bandwidth. Hence, doubling the oversampling ratio is equivalent to increasing the quantizer levels by a half-bit as far as the quantization noise is concerned.

Oversampling allows the use of a lower-resolution converter without sacrificing noise performance. Another way of looking at this effect is that by doubling the sampling rate, we now have two correlated signal samples; thus, the signal power is increased by 6 dB. However, having two noise samples only doubles the noise power, so the noise power is increased by only 3 dB. In other words, the signal samples are correlated; the noise samples are not. Thus, there is an SNR improvement of 3 dB that corresponds to a half-bit resolution improvement. Oversampling

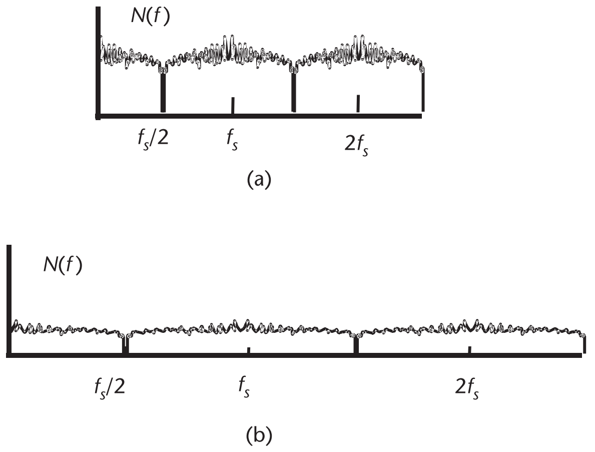


Figure 9.4 Quantization noise reduction by increasing OSR: (a) low sampling frequency, and (b) higher sampling frequency.

has the advantage that it eases requirements on analog antialiasing filters for A/D converters or deglitch filters for D/A converters because oversampling results in wide transition bands; hence, only low-order filters are needed. However, the higher sampling rate means the digital circuits have to run faster, and that consumes more power. It also means that the signal bandwidth has to be much lower than the sampling rate, which limits the conversion rate.

Example 9.1: Oversampling Voice with a Video A/D Converter
Let us assume that we are sampling a 4-kHz voice signal with an 8-bit video A/D at 8 MHz. Determine the oversampling ratio and the improvement in SNR.

Solution: The Nyquist rate is 8 kHz; thus, the oversampling ratio is 1,000. This will result in $0.5 \log_2(1{,}000) = 5$ bits extra resolution. Since we started with 8 bits, the OSR of 1,000 provides a total of 13-bit accuracy.

Is quantization really like noise? Consecutive 8-MHz samples of a voice signal would be highly correlated; as a result, there could also be correlation in the quantization error. Consequently, quantization power would not be reduced by averaging. For samples where noise is correlated from sample to sample, one can use dithering (adding a small random signal) in $\Sigma\Delta$ modulation to randomize the correlated quantization noise power. Integral (large-scale) nonlinearity is also not improved by oversampling. In frequency synthesis, any correlation between noise samples in the time domain generates spurious tones (spurs) in the frequency domain. In contrast to white noise, the spurs are located at particular frequencies that cannot be reduced by averaging when the oversampling ratio increases. Thus, the oversampling technique has no impact on fractional spurious tones, although it can reduce random white noise. In frequency synthesis, randomization techniques are needed to break the correlation and repeated patterns and, thus, to spread the spur energy over the frequency band. The randomization process results in a quasiwhite noise characteristic that can then be reduced by oversampling techniques. In conclusion, oversampling trades speed for resolution and allows the use of high-speed digital circuits instead of precise analog circuits, resulting in design simplicity and a certain amount of design freedom.

9.2.2 Noise-Shaping Effect

While oversampling can reduce the random quantization noise by averaging it over a wider sampling bandwidth, another useful scheme for quantization noise reduction is *noise shaping* using feedback. Consider a feedback model shown in Figure 9.5. In this figure, an additive noise model of the quantizer follows a filter with transfer function of $H(s)$. Then, negative feedback is added to stabilize the system.

From control theory, the output is given by

$$Y(s) = \frac{H(s)}{1 + H(s)} X(s) + \frac{1}{1 + H(s)} E(s) \tag{9.13}$$

If $|H(s)| \gg 1$, such as when the system contains an integrator and only low-frequency components are considered, the output $Y(s)$ due to the input $X(s)$, the so-called signal transfer function, can be approximated as

$$Y(s) = \frac{H(s)}{1 + H(s)} X(s) \approx X(s) \tag{9.14}$$

Thus, the signal transfer function is unity at low frequencies. Similarly, the output due to the noise $E(s)$, the noise transfer function (NTF), can be approximated by

$$\mathrm{NTF}(s) = \frac{1}{1 + H(s)} E(s) \approx 0 \tag{9.15}$$

It is apparent that the output quantization noise can be reduced in certain frequency bands if $H(s)$ is designed to have high gain in those frequency bands. The most troublesome noise is the close-in quantization noise since it is hard to remove using a LPF. In an open-loop oversampling system without noise-shaping feedback, as shown in Figure 9.6, the quantization noise is a white noise uniformly distributed from $0 \le f \le f_s/2$ with noise power density of $\Delta^2/6f_s$. In a feedback oversampling system with a lowpass transfer function for $H(s)$, the in-band quantization noise can be shaped as illustrated in Figure 9.7. The resultant noise transfer function of $1/[1 + H(s)]$ has a highpass effect. Hence, the quantization noise is

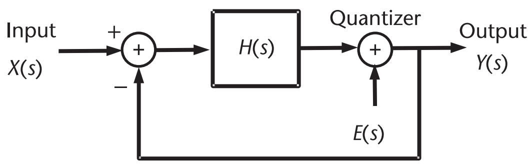


Figure 9.5 Feedback model of an oversampling noise-shaping system.

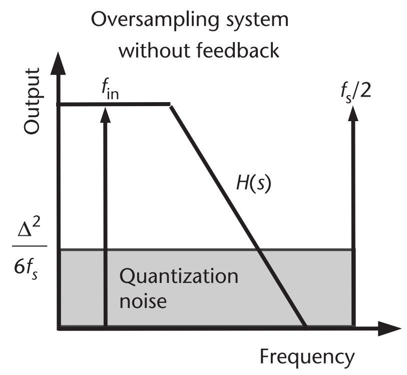


Figure 9.6 Output of an oversampling system without feedback.

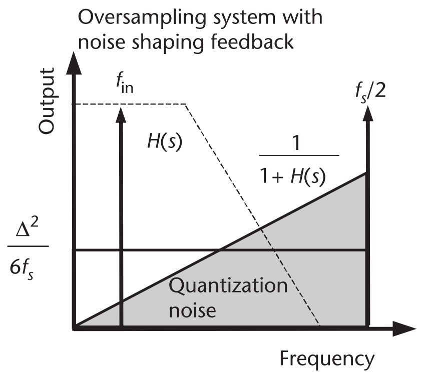


Figure 9.7 Output of an oversampling system with noise-shaping feedback.

highpass-shaped by the feedback loop, leading to lower in-band noise. Note that the total quantization noise energy for both systems with and without feedback is the same. In other words, the total shaded area in each of Figures 9.6 and 9.7 is $\Delta^2/12$. The only difference is that the quantization noise with feedback is shifted to the higher frequency band, where it can be easily filtered. The following sections provide detailed analysis of various noise-shaping schemes.

Oversampling and noise shaping are two related techniques. *Oversampling* refers to sampling beyond the Nyquist rate. *Noise shaping* refers to shaping the noise spectrum such that the noise is lowered in the signal band and increased outside of the signal band, where the noise can be easily removed. In particular, the noise shaping in frequency synthesis employs a $\Sigma\Delta$ modulator to form a highpass filter such that the spurs and noise can be shifted to a higher frequency band, where they can be removed easily by a lowpass loop filter. Noise shaping is most useful when it is done in combination with oversampling. As will be seen in the following sections, $\Sigma\Delta$ modulation greatly enhances the oversampling effect by using feedback systems. The oversampling system with $\Sigma\Delta$ modulation is often called a high-order oversampling system since the conventional oversampling system without $\Sigma\Delta$ modulation is, in fact, a zero-order feedback system.

9.2.3 An Overview of ΣΔ Modulators

To introduce the concept of a ΣΔ converter, let us start with a delta modulator, as illustrated in Figure 9.8, where the error signal v_e to be sampled by the quantizer is the difference, or *delta*, between the input signal and the feedback signal. The feedback signal is the integral of the output. Thus, if the feedback signal, v_x, is less than the input signal, x, the output signal y goes high, which brings v_x towards x.

Because a delta modulator will not shape quantization noise, a more useful implementation is to move the integrator into the forward path after the difference block. This configuration is called a ΣΔ modulator, as shown in Figure 9.9. In this circuit, the error signal is first integrated and then passed through a quantizer before it is fed back to the input. Since the integrator has a high gain at low frequency, the error voltage at low frequency is small. A small error at low frequency means that the low-frequency component of the output bit stream is equal to the input signal. Thus, the low-frequency component can be recovered using a lowpass filter or an integrator. In general, the output pulse density represents the input voltage (i.e., the low-frequency part of this waveform follows the input signal).

A ΣΔ modulator is sometimes called a ΔΣ modulator, and people have debated the name in accordance with their perception of the order in which the operations take place. *Sigma* refers to integration, while *delta* refers to the comparison. In this

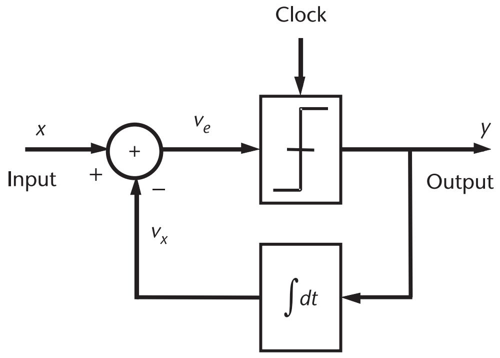


Figure 9.8 A delta modulator.

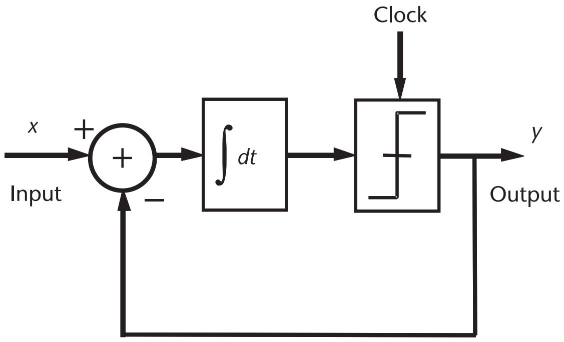


Figure 9.9 A ΣΔ modulator.

book, the convention ΣΔ will be used arbitrarily. Since the clock is running at a rate much higher than the input signal rate, this system is an *oversampling system*.

9.2.4 First-Order ΣΔ Modulators

One of the most basic forms of the ΣΔ modulator is a first-order ΣΔ modulator, introduced in Figure 9.9. It is a more efficient oversampling A/D converter compared to a simple quantizer without feedback. The analysis will be started by assuming that the modulator contains a multilevel uniform quantizer with unity gain. The integrator averages the signal fed to the quantizer input. The quantized output y is fed back and subtracted from the input x. Thus, feedback forces the average value of the quantized output signal to track the average input.

The first-order ΣΔ modulator will be analyzed using the equivalent circuit shown in Figure 9.10, where the quantizer is represented as an additive quantization error e_i at the time step i with the gain Δ set to unity. Because this is a sampled-data circuit, we represent the integration using an accumulator. It can be easily derived that the output of the accumulator is

$$w_i = \frac{z^{-1}}{1 - z^{-1}} v_i \tag{9.16}$$

With some manipulation, and remembering that z^{-1} is a unit delay,

$$w_i = v_{i-1} + w_{i-1} \tag{9.17}$$

Now the output from the ΣΔ modulator is

$$y_i = w_i + e_i = v_{i-1} + w_{i-1} + e_i \tag{9.18}$$

We note that

$$v_{i-1} = x_{i-1} - y_{i-1} \tag{9.19}$$

and that

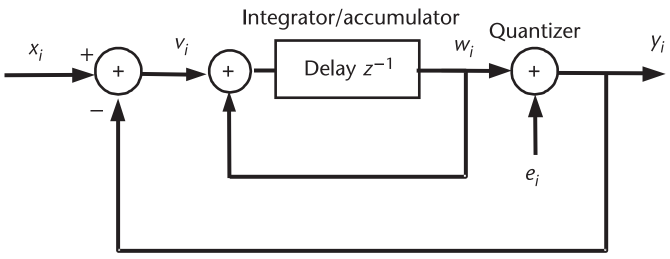


Figure 9.10 The equivalent circuit of the first-order ΣΔ modulator.

$$y_{i-1} = w_{i-1} + e_{i-1} \qquad (9.20)$$
$$w_{i-1} = y_{i-1} - e_{i-1}$$

Therefore, substituting (9.20) and (9.19) into (9.18), the quantized signal with the first-order ΣΔ modulator is given by

$$y_i = x_{i-1} - y_{i-1} + y_{i-1} - e_{i-1} + e_i \qquad (9.21)$$
$$y_i = x_{i-1} + e_i - e_{i-1}$$

Note that the error now becomes the difference between the errors obtained in the consecutive sampling steps. Thus, the first-order ΣΔ modulator differentiates the quantization error. In other words, it modulates the quantization noise as the first-order difference of the quantization error. The signal, however, is left unchanged, except for a propagation delay that is acceptable in most systems.

The same system can also be analyzed in the frequency domain. The accumulator output can be represented as

$$Y(z) = W(z) + E(z) = V(z)\frac{z^{-1}}{1 - z^{-1}} + E(z) \qquad (9.22)$$
$$Y(z) = [X(z) - Y(z)]\frac{z^{-1}}{1 - z^{-1}} + E(z)$$

After some manipulation, the quantizer output can be found as

$$Y(z) = z^{-1}X(z) + (1 - z^{-1})E(z) \qquad (9.23)$$

Referring to the basic control theory given in Appendix A, we recall that $(1 - z^{-1})$ represents a highpass filter with a zero at $z = 1$ and a pole at $z = 0$. Thus, the first-order ΣΔ modulator highpass filters the quantization noise while leaving the signal unchanged, except for a propagation delay of one sampling cycle. The name "ΣΔ noise shaping" originated from its property of shifting the noise to a higher frequency due to its highpass filter characteristic. In this way, using a ΣΔ modulator in a fractional-*N* frequency synthesizer, spurious components can be shifted to higher frequencies, where a lowpass loop filter can remove them.

Assuming that the input signal is sufficiently busy, the quantization error can be treated as white noise, which is uncorrelated with the signal. The noise spectral density of the first-order ΣΔ modulator $n_i = e_i - e_{i-1}$ can be shown as

$$N_1(f) = E(z)(1 - z^{-1}) = E(f)\left|1 - e^{-j2\pi fT}\right| = 2e_{\text{rms}}\sqrt{2T}\,\sin(\pi fT) \qquad (9.24)$$

where $E(f)$ is defined in (9.6), and e_{rms}^2 is defined in (9.5). Therefore, the noise power in the signal band is calculated as

$$n_0^2 = \int_0^{f_0} |N_1(f)|^2 \, df \approx e_{\text{rms}}^2 \frac{\pi^2}{3} (2f_0 T)^3 \tag{9.25}$$

where we have assumed a sufficiently high OSR such that $f_s^2 \gg f_0^2$ and $\sin(\pi f T) \approx \pi f T$. The rms noise magnitude is given by

$$n_0 = e_{\text{rms}} \frac{\pi}{\sqrt{3}} (2f_0 T)^{3/2} = e_{\text{rms}} \frac{\pi}{\sqrt{3}} (\text{OSR})^{-3/2} \tag{9.26}$$

Thus, in a sampled system with the first-order ΣΔ modulator, each doubling of the oversampling ratio reduces the quantization noise by 9 dB and increases the effective bits by 1.5. Recall that in a sampled system without any feedback (a zero-order system), each doubling of the oversampling ratio reduces the quantization noise by only 3 dB and increases the effective bits by 0.5. Although the ΣΔ effect strengthens the oversampling effect, the two effects originate from different phenomena. The feedback with ΣΔ modulation reduces the in-band noise by shaping the noise away from the signal band, while the oversampling effect reduces the in-band noise by averaging the noise power over a wider bandwidth.

9.2.5 Second-Order ΣΔ Modulators

As shown, a ΣΔ feedback via integration shapes the spectrum of the modulation noise by placing most of its energy outside the signal band. The ΣΔ noise-shaping effect is determined by its highpass characteristic. This section will show that the shape of the noise spectrum is determined by the order of the ΣΔ modulator. Higher-order ΣΔ modulators can be configured in various ways. By adding another accumulator and feedback loop to the first-order ΣΔ modulator, a second-order ΣΔ modulator can be constructed, as shown in Figure 9.11.

With some mathematical manipulations, the output of this modulator can be expressed as

$$y_i = x_{i-1} + (e_i - 2e_{i-1} + e_{i-2}) \tag{9.27}$$

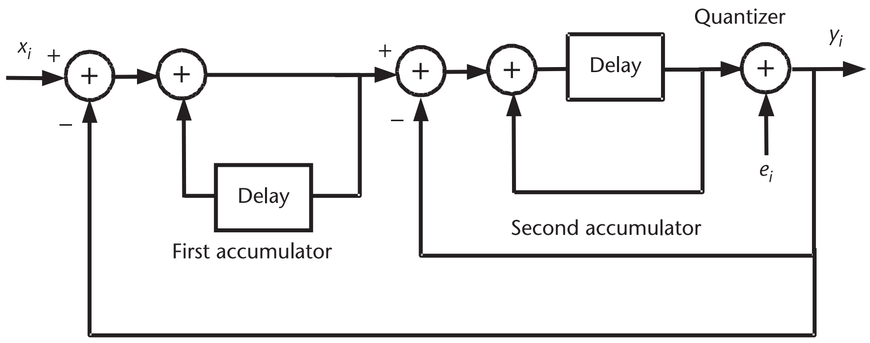


Figure 9.11 The equivalent circuit of a second-order ΣΔ modulator.

and its z domain representation is

$$Y(z) = z^{-1}X(z) + (1 - z^{-1})^2 E(z) \tag{9.28}$$

Note that the modulator output noise is now the second-order difference of the quantization error. The noise spectral density of the second order ΣΔ modulator is given by

$$N_2(f) = E(f)(1 - e^{-j\omega T})^2 = 4e_{\text{rms}}\sqrt{2T}\,\sin^2(\pi f T) \tag{9.29}$$

Assuming again that $f_s \gg f_0$, the rms noise magnitude in the signal band can be found by integrating the noise spectral density as

$$n_0 = \left(\int_0^{f_0} |N(f)|^2\, df\right)^{1/2} = e_{\text{rms}}\frac{\pi^2}{\sqrt{5}}(2f_0T)^{5/2} = e_{\text{rms}}\frac{\pi^2}{\sqrt{5}}(\text{OSR})^{-5/2} \tag{9.30}$$

Notice that the noise is now lowered by 15 dB for every doubling of the oversampling ratio and, thus, achieves 2.5 extra bits of resolution.

9.2.6 High-Order ΣΔ Modulators

The previous analysis can be extended to a ΣΔ modulator of any order. In general, an nth-order ΣΔ modulator can be configured by adding more feedback loops to the circuit, as shown in Figure 9.12. When there are n loops, and the system is stable, its output can be expressed in the z domain as

$$Y(z) = z^{-1}X(z) + (1 - z^{-1})^n E(z) \tag{9.31}$$

and its noise spectral density can be found as

$$|N_n(f)| = e_{\text{rms}}\sqrt{2T}\,[2\sin(\pi f T)]^n \tag{9.32}$$

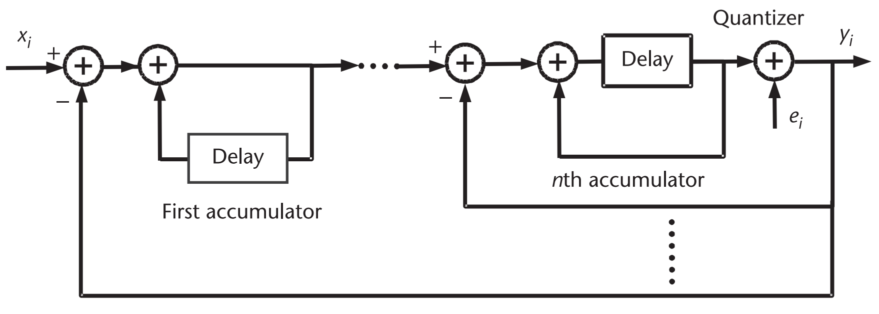


Figure 9.12 An equivalent circuit of an *n*th-order ΣΔ modulator.

Figure 9.13 plots $N(f)$ for different orders of ΣΔ modulators with OSR = 32. As shown, in a pure oversampling system without a ΣΔ modulator (zero-order ΣΔ), noise spectral density $N(f)$ is a random white noise uniformly spread out from 0 to $f_s/2$. For an oversampling system with a ΣΔ modulator, $N(f)$ is shaped such that the in-band noise at lower frequencies is reduced compared to the original white noise floor, while the out-of-band noise at higher frequencies is increased. However, the ΣΔ modulator does not affect the total noise power over the band from 0 to $f_s/2$. Readers can verify that the integral of $|N_n(f)|^2$ over the range from 0 to $f_s/2$ (i.e., the areas under different curves in Figure 9.13) should be a constant. The difference for various orders of ΣΔ modulators is the slope of the noise-shaping curve. According to the highpass transfer function given in (9.31), the noise-shaping slope of an nth-order ΣΔ modulator is $n \times 20$ dB/dec, that is, 20 dB/dec for the first-order ΣΔ modulator and 60 dB/dec for a third-order ΣΔ modulator, as demonstrated in Figure 9.13. This rule is also valid for an oversampling system without ΣΔ feedback if we treat it as the case where $n = 0$.

For large oversampling ratios (OSR > 2), the in-band rms noise magnitude is given by

$$n_0 = e_{\text{rms}} \frac{\pi^n}{\sqrt{2n+1}} (2f_0 T)^{n+(1/2)} = e_{\text{rms}} \frac{\pi^n}{\sqrt{2n+1}} \left(\frac{1}{\text{OSR}}\right)^{n+(1/2)} \tag{9.33}$$

where it is once more assumed that $f_s \gg f_0$, just as in (9.26) and (9.30). Thus, for an nth-order ΣΔ modulator, it can be concluded that the in-band rms noise falls by $3(2n + 1)$ dB for every doubling of the oversampling ratio, resulting in $n + 0.5$

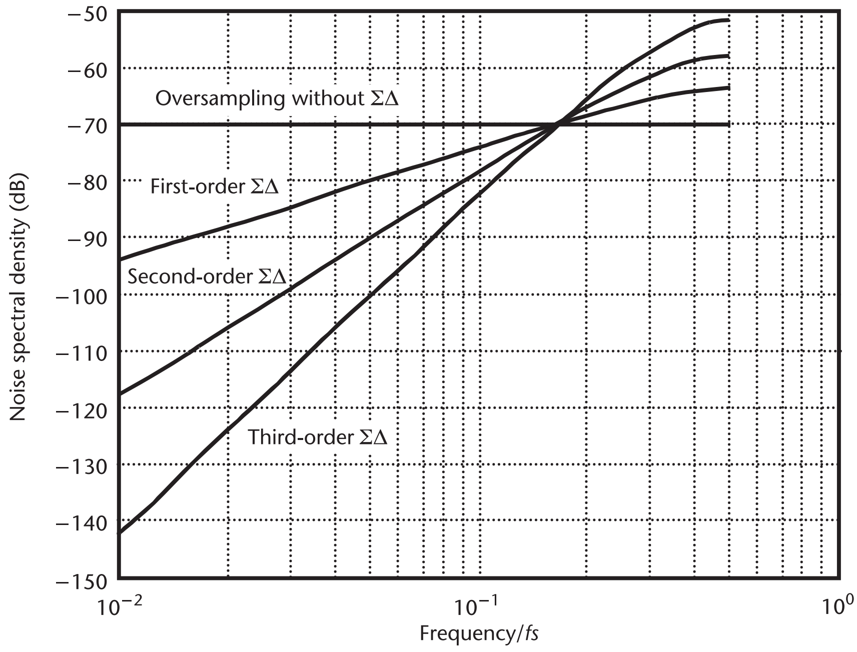


Figure 9.13 Noise spectral density $N(f)$ for different order ΣΔ modulators with OSR = 32. The noise-shaping slope of an nth-order ΣΔ modulator is $n \times 20$ dB/dec.

extra bits of resolution. This generic conclusion is valid for both ordinary sampled systems without feedback (zero-order feedback with $n = 0$) and a sampled system with ΣΔ feedback. In the expression, the 3 dB (0.5 effective bits) is due to the oversampling effect in an ordinary sampled system without feedback, while the $6n$ dB (n effective bits) is due to the ΣΔ noise-shaping effect. Figure 9.14 plots the in-band rms noise versus OSR for different orders of ΣΔ modulators and clearly demonstrates the noise-reduction effect discussed above. It is evident that adding ΣΔ feedback strengthens the oversampling effect. On the other hand, the ΣΔ modulator will not greatly benefit the noise reduction if the OSR is not sufficiently high. The ΣΔ noise shaping will increase the noise, rather than reduce it, when OSR is less than two.

The performance improvement due to oversampling and noise shaping can be characterized by the number of *effective bits*, which leads to the same SNR over a fixed bandwidth. The SNR with oversampling and noise shaping can be found, considering signal power given in (9.10) and noise power given in (9.33) as

$$\mathrm{SNR} = 10\log\left[\frac{\frac{1}{8}(2^N-1)^2\Delta^2}{\frac{\Delta^2}{12}\,\frac{\pi^{2n}}{2n+1}\left(\frac{1}{\mathrm{OSR}}\right)^{2n+1}}\right] \tag{9.34}$$

$$\approx 10\log\left[\frac{3}{2}\cdot 2^{2N}\cdot \mathrm{OSR}^{2n+1}\cdot\left(\frac{\pi^{2n}}{2n+1}\right)^{-1}\right]$$

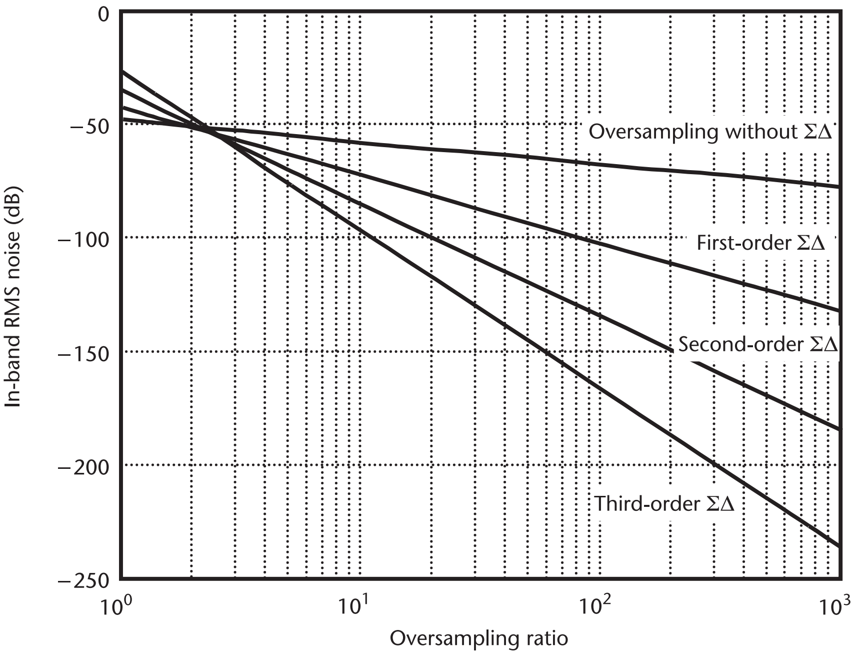


Figure 9.14 The in-band rms noise versus OSR for different order ΣΔ modulators. The in-band noise falls $3(2n + 1)$ dB for every doubling of OSR for an nth-order ΣΔ modulator.

Further manipulation of the above expression yields

$$\mathrm{SNR} = 6.02N + 1.76 + 3(2n + 1)\log_2 \mathrm{OSR} - 10\log_{10}\left(\frac{\pi^{2n}}{2n + 1}\right) \quad (9.35)$$

where N is the number of quantizer bits, and n is the order of the ΣΔ modulator. The lowered quantization noise due to oversampling and noise shaping leads to an effectively increased number of quantizer bits:

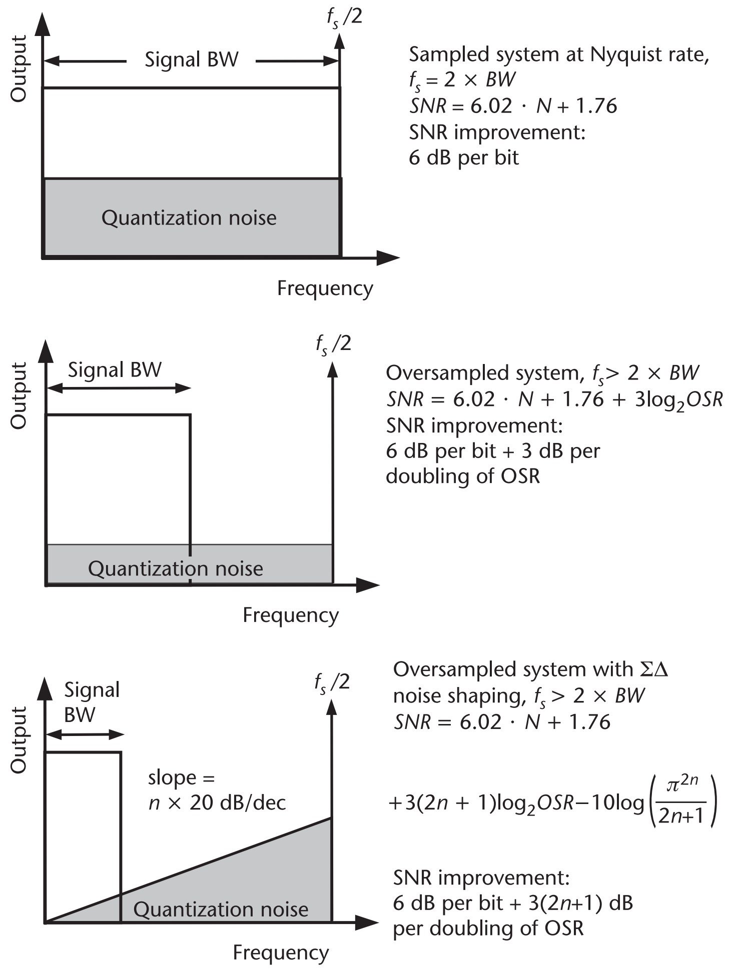
$$N_{\mathrm{effective}} \approx N + \frac{2n + 1}{2}\log_2 \mathrm{OSR} - 1.66 \cdot \log_{10}\left(\frac{\pi^{2n}}{2n + 1}\right) \quad (9.36)$$

Note that the number of effective bits cannot be improved greatly without significant oversampling. At the Nyquist rate (OSR = 1), there is even a reduction in the number of effective bits due to the last term in (9.36) introduced by the ΣΔ modulator. With the above analysis, the concepts of oversampling and noise shaping are further compared and illustrated in Figure 9.15. As shown, when the signal bandwidth approaches $f_s/2$ (the Nyquist rate), there is no benefit from noise shaping.

9.3 ΣΔ Modulation in Fractional-*N* Frequency Synthesis

The output of a fractional-*N* synthesizer is related to a reference frequency by a rational divisor, which is obtained by periodically toggling the dual-modulus or multimodulus dividers. However, switching between different divisor values results in undesirable phase jitter or spurs near the desired carrier frequency. A ΣΔ modulator can move the in-band noise to higher frequencies utilizing its highpass noise characteristic. The ΣΔ modulator forms a highpass filter at the fractional accumulator output, where the fractional spurs are created, moving the close-in fractional spurs to higher frequencies where they can be removed by the loop filter. A ΣΔ fractional-*N* synthesizer is conceptually sketched in Figure 9.16, leaving the ΣΔ fractional accumulator as a black box, which the following sections investigate in detail.

High-order ΣΔ noise shaping is important when a large accumulator size is used. As discussed in Chapter 2, the fractional accumulator shown in Figure 9.16 with size F and an input word of K will generate K carry-out pulses for every F clock cycles. The accumulator is clocked at f_r/R when the loop is in lock. Thus, the accumulator carry-out has a repeated pulse pattern at a rate of $(f_r/R \cdot K/F)$. The fractional accumulator periodically generates the carry-out that toggles the loop division ratio. Recall that any repeated pattern in the time domain causes spurious tones in the frequency domain. The fractional spurious components associated with periodically toggling the loop division ratio are hence expected at a multiple of the carry-out frequency with the spur closest to the carrier located at $(f_r/R \cdot K/F)$. When $K = 1$, the closest fractional spur occurs at $(f_r/R \cdot 1/F)$. Therefore, the larger the accumulator size, the closer the fractional spur is to the



$f_s/2$
Output
Signal BW
Sampled system at Nyquist rate,
$f_s = 2 \times BW$
$SNR = 6.02 \cdot N + 1.76$
SNR improvement:
6 dB per bit
Quantization noise
Frequency
$f_s/2$
Output
Signal BW
Oversampled system, $f_s > 2 \times BW$
$SNR = 6.02 \cdot N + 1.76 + 3\log_2 OSR$
SNR improvement:
6 dB per bit + 3 dB per doubling of OSR
Quantization noise
Frequency
Output
Signal BW
$f_s/2$
Oversampled system with ΣΔ noise shaping, $f_s > 2 \times BW$
$SNR = 6.02 \cdot N + 1.76$
slope = $n \times 20$ dB/dec
$+3(2n+1)\log_2 OSR - 10\log\left(\frac{\pi^{2n}}{2n+1}\right)$
SNR improvement:
6 dB per bit + 3(2n+1) dB per doubling of OSR
Quantization noise
Frequency

Figure 9.15 Comparison of oversampling and noise-shaping effects.

carrier, and a higher-order ΣΔ modulator is needed for spur reduction. The order of the ΣΔ can be selected based upon the system noise requirement.

Two types of ΣΔ modulators have been used in fractional-*N* frequency synthesis [3–6]. One is a single-loop modulator, and another includes cascaded accumulators called multistage noise shaping (MASH), also sometimes called multiloop ΣΔ modulators [5]. The single-loop modulator can have a single-bit or multibit output, depending on the desired quantization noise floor, while the multiloop MASH architecture normally has a single carry-out bit per accumulator and, thus, always has a multibit output. This discussion will begin with a popular architecture using a multiloop ΣΔ modulator, originally proposed by Wells [3] and Miller [4]. In fractional-*N* frequency synthesis, the instantaneous divisor value, which is achieved using integer frequency dividers, is always off the desired fractional value; yet, the time-averaged integer divisor value is equal to the desired fractional value and, therefore, never changes when the loop is in lock. A simple fractional-*N* scheme achieves the fractional divisor value by toggling the loop divisor in a repeated pattern (e.g., toggling the divisor between two integer numbers). A ΣΔ modulator

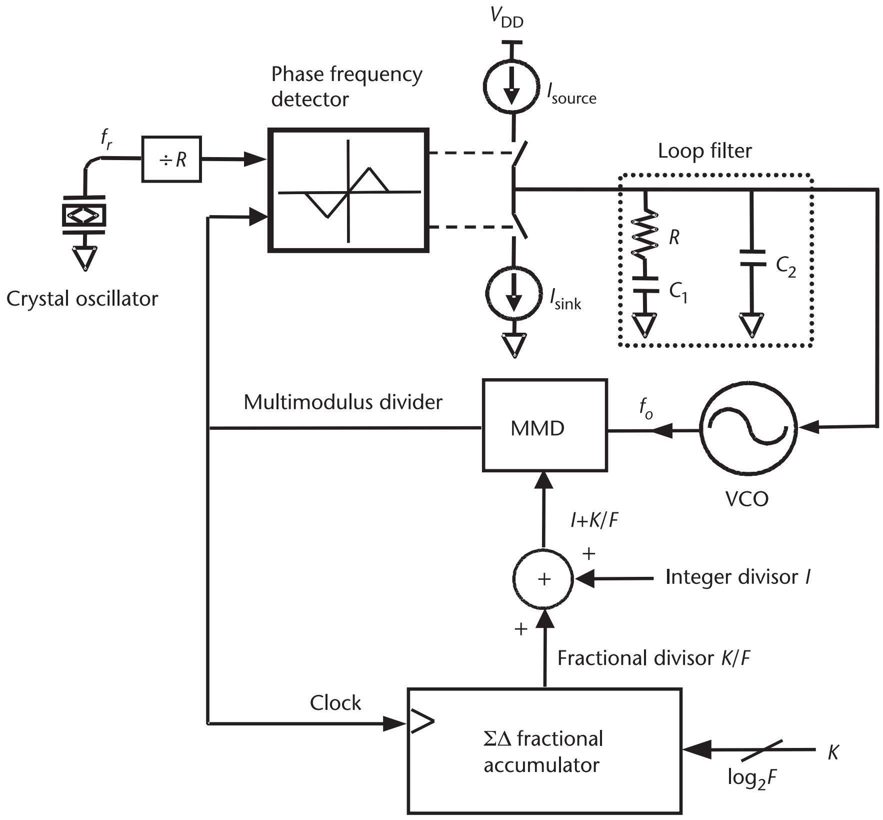


Figure 9.16 Fractional-*N* frequency synthesizer with a ΣΔ modulator.

is applied to break the repeated temporal (time domain) pattern of the fractional-*N* scheme and, thus, reduces the spectral spurs. Like the simple fractional-*N* scheme, any ΣΔ architecture should provide a zero net change in divisor value, meaning that the time-averaged divisor value should not be altered. A ΣΔ based fractional-*N* synthesizer dithers the divisor value following a specific temporal pattern, which corresponds to modulating the fractional-*N* accumulator output with a highpass characteristic in the frequency domain. Thereby, the fractional spurs resulting from the altering of the integer divisor value are moved to higher frequencies, where they can be attenuated much more effectively by the loop lowpass filter.

9.3.1 A First-Order ΣΔ Modulator for Fractional-*N* Frequency Synthesis

The noise-shaping concept introduced for oversampling data converters can also be applied to fractional-*N* synthesis. In this section, the ΣΔ modulator discussed in Section 9.2.4 is applied to this problem. In frequency synthesis, the desired fractional frequency is equivalent to the analog input of an A/D converter. The digital modulus control of the dual-modulus prescaler or the multimodulus divider is analogous to the 1-bit or multibit quantizer output. Let us investigate a ΣΔ modulator like those previously considered, where the input is an analog signal, and the output is a 1-bit sampled digital data stream, as shown in Figure 9.17(a).

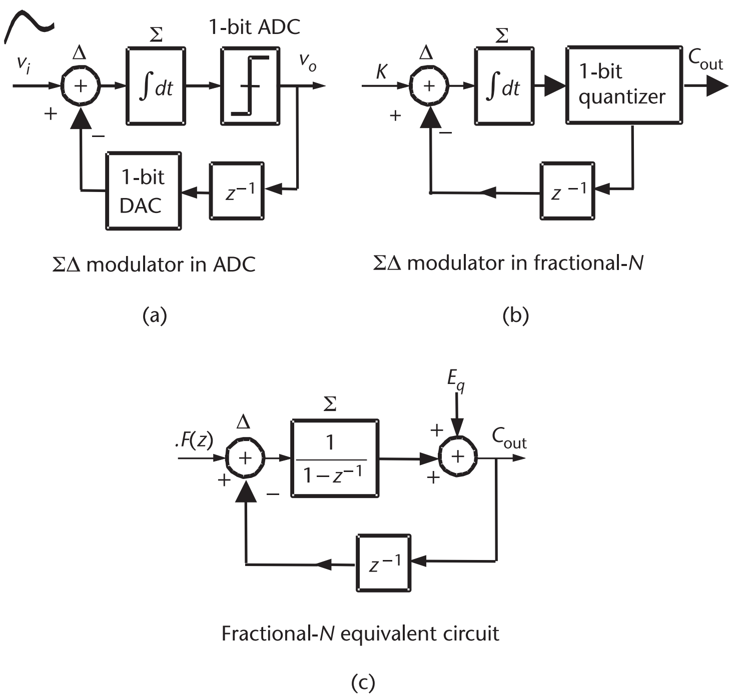


Figure 9.17 Conversion of a first-order ΣΔ modulator in an ADC to a first-order ΣΔ modulator in a fractional-*N* accumulator: (a) modulator for an ADC; (b) modulator in a fractional-*N* accumulator; and (c) fractional-*N* equivalent circuit.

In fractional-*N* synthesis, the fractional spurs come from the fractional accumulator, as discussed in Chapter 2, where the input is a constant frequency word *K* in digital format, and the output is the 1-bit carry-out used to control the modulus of the divider. Adding ΣΔ blocks to a fractional-*N* accumulator, we obtain a ΣΔ modulated accumulator, as shown in Figure 9.17(b). The integrator carries out the accumulator function and has a transfer function of $1/(1 - z^{-1})$ when the delay unit is placed in the feedback path. A delay unit can also be combined with the integrator to form a transfer function of $z^{-1}/(1 - z^{-1})$, as was done previously.

The 1-bit comparator and the 1-bit quantizer are equivalent to the carry-out of the accumulator; that is, the 1-bit quantizer truncates the MSB of the integrator output and discards the rest of the bits. Since the input is a digital frequency word, the 1-bit DAC is not necessary in fractional-*N* synthesis. Converting to the sampled time domain and replacing the quantizer with its quantization noise model, an equivalent circuit is obtained, as shown in Figure 9.17(c), where $.F(z)$ denotes the fractional part of the divisor value, which is equivalent to K/F in the frequency domain. Note that the "." preceding the "$F(z)$" is part of the function name. Although the input to the fractional accumulator is a constant (K/F), defining the input to the ΣΔ modulator as a function in the z domain is generally necessary when multiple accumulators are cascaded to form a higher-order modulator, where the inputs to all the accumulators, except for the first one, are time-varying.

If the integrator has a z transfer function of $1/(1 - z^{-1})$ with the delay unit in the feedback path, the ΣΔ fractional-*N* outputs can be found as

$$C_{\text{out}}(z) = \frac{\dfrac{1}{1-z^{-1}}}{1+\dfrac{z^{-1}}{1-z^{-1}}}.F(z) + \frac{1}{1+\dfrac{z^{-1}}{1-z^{-1}}}E_q(z) = .F(z) + (1-z^{-1})E_q(z) \tag{9.37}$$

where E_q is the quantization error introduced when truncating the accumulator value to a 1-bit carry-out. Note that this equation is almost the same as (9.23), as expected. It is evident that the quantization error E_q is highpass filtered by means of adding the first-order ΣΔ modulator, while the fractional frequency word $.F(z)$ is not affected. Thus, the spurs due to the periodic truncation are moved to higher frequencies, and the average divisor value is left unchanged, as desired. Just as it is possible to build high-order ΣΔ modulators for data converters, it is also possible to build high-order ΣΔ modulators for fractional-*N* synthesis. The techniques for adding feedback loops are very similar for both styles of modulators.

9.3.2 MASH ΣΔ Modulator

Figure 9.18 shows a MASH ΣΔ modulator, comprising three loops. The three-loop ΣΔ topology shown in Figure 9.18 is also called a MASH 1-1-1 structure because it is a cascaded ΣΔ structure with three first-order loops. Each of the three

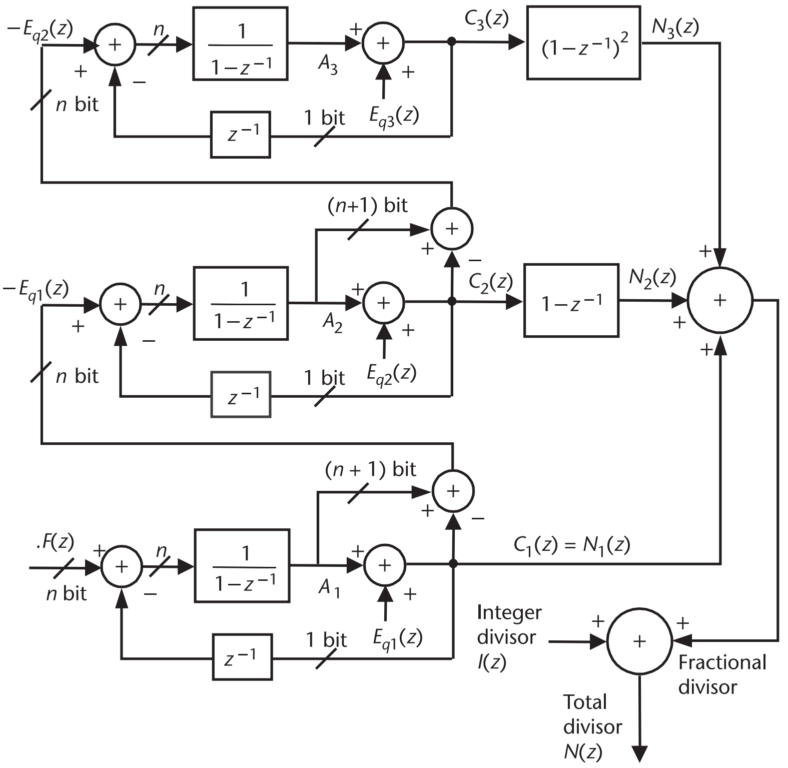


Figure 9.18 A triple-loop MASH 1-1-1 ΣΔ modulator for fractional-*N* synthesis.

loops is identical to the single-loop architecture discussed previously. The input of the second loop is taken from the quantized error E_{q1} of the first loop, while the input of the third loop is taken from the quantized error E_{q2} of the second loop. Thus, only the first loop has a constant input, which is the fractional portion of the desired rational divide number $.F(z)$ (i.e., the fine-tune frequency word). The integer part of the frequency word $I(z)$, the coarse-tune frequency word, is added at the output of the triple-loop ΣΔ modulator. Thus, $N_{\text{div}}(z) = I(z) + .F(z)$ is the time sequence used to control the divider ratios. Differentiator blocks with transfer functions of $(1 - z^{-1})$ and $(1 - z^{-1})^2$ are added at the output of the second and third loops, respectively. Their function will become evident from the following mathematical derivations. The modulator is clocked at the reference frequency, reflecting the sampled nature of the circuit. The z domain transfer functions for the triple-loop ΣΔ modulator can be derived as

$$\begin{aligned} N_1(z) &= C_1(z) \\ &= \frac{\frac{1}{1-z^{-1}}}{1+\frac{z^{-1}}{1-z^{-1}}}.F(z) + \frac{1}{1+\frac{z^{-1}}{1-z^{-1}}}E_{q1}(z) \\ &= .F(z) + (1-z^{-1})E_{q1}(z) \end{aligned} \tag{9.38}$$

$$\begin{aligned} N_2(z) &= (1-z^{-1})C_2(z) \\ &= (1-z^{-1})[-E_{q1}(z) + (1-z^{-1})E_{q2}(z)] \\ &= -(1-z^{-1})E_{q1}(z) + (1-z^{-1})^2E_{q2}(z) \end{aligned} \tag{9.39}$$

$$\begin{aligned} N_3(z) &= (1-z^{-1})^2C_3(z) \\ &= (1-z^{-1})^2[-E_{q2}(z) + (1-z^{-1})E_{q3}(z)] \\ &= -(1-z^{-1})^2E_{q2}(z) + (1-z^{-1})^3E_{q3}(z) \end{aligned} \tag{9.40}$$

As shown, the first loop generates the fractional divisor value $.F(z)$ with the byproduct of quantization error E_{q1}, which is fed to the input of the second loop for further processing. The second loop cancels the previous loop's quantization error E_{q1} by the differential block $(1 - z^{-1})$ in its output path. The only quantization noise term left after summing the first- and second-loop outputs is the quantization error E_{q2}, which is second-order noise shaped. When this noise term is further fed to the input of the third loop, the loop generates a negative noise term to cancel the previous loop's quantization error E_{q2} by the second-order differential block $(1 - z^{-1})^2$ in its output path. Summing the outputs of the three loops, we obtain the modulated divisor value as

$$N(z) = I(z) + N_1(z) + N_2(z) + N_3(z) = I(z) + .F(z) + (1-z^{-1})^3E_{q3}(z) \tag{9.41}$$

where $I(z)$ and $.F(z)$ are the integer portion and the fractional portion of the division ratio, respectively. As desired, the fractional divisor value $.F(z)$ is not affected by the modulator, while the quantization error generated in the last loop E_{q3} is noise shaped by a third-order highpass function of $(1 - z^{-1})^3$. The quantization error generated in the first and second loops are totally canceled. As a result, the total quantization noise is equal to that of a single loop, although three loops are used. Therefore, the multiloop ΣΔ architecture provides high-order noise shaping without additional quantization noise.

In general, the output frequency of a fractional-*N* synthesizer with a cascaded m-loop ΣΔ modulator can be expressed as

$$f_0(z) = [I(z) + .F(z)]f_r + (1 - z^{-1})^m E_{qm}(z) f_r \tag{9.42}$$

where $N(z) = I(z) +. F(z)$ is the desired division ratio, and the second term is the frequency noise due to the fractional spurs.

It is interesting to note that the coefficients of the MASH highpass noise transfer function $(1 - z^{-1})^m$ follow the successive rows in Pascal's triangle:

$$\begin{array}{ccccccccccc} & & & & & 1 & & & & & \\ & & & & 1 & & -1 & & & & \\ & & & 1 & & -2 & & 1 & & & \\ & & 1 & & -3 & & 3 & & -1 & & \\ & 1 & & -4 & & 6 & & -4 & & 1 & \\ 1 & & -5 & & 10 & & -10 & & 5 & & -1 \end{array} \tag{9.43}$$

For example, the third-order noise transfer function of $(1 - z^{-1})^3$ can be expanded as $1 - 3z^{-1} + 3z^{-2} - z^{-3}$, where the coefficients follow the fourth row of Pascal's triangle.

Example 9.2: Comparison of a MASH 1-1-1 ΣΔ Modulator with a Simple Fractional-N Accumulator

Compare the fractional accumulator output ($m = 1$) to the third-order ΣΔ modulator output ($m = 3$) for the case of $N(z) = I(z) +. F(z) = 100 + 1/32$, with a reference frequency of $f_r = 10$ MHz.

Solution: The simulated accumulator outputs are given in Figures 9.19 and 9.20 for $m = 1$ and $m = 3$. As shown, the fractional accumulator without ΣΔ modulator ($m = 1$) has a carry-out in every 32 f_r cycles, which forces the loop divider to divide by 100 for 31 cycles and then to divide by 101 for 1 cycle. The periodic phase-correction pulse due to dividing by 101 generates fractional spurs with a uniform spacing of $f_r/32 = 312.5$ kHz, as shown in Figure 9.20(a). If three cascaded loops are used, the ΣΔ accumulator outputs are dithered around the correct value, as shown in Figure 9.19(b). Note that the ΣΔ noise shaper breaks the periodicity of the fractional divisor sequences. In the frequency domain, the discrete spurs become more random with their energy pushed towards the higher

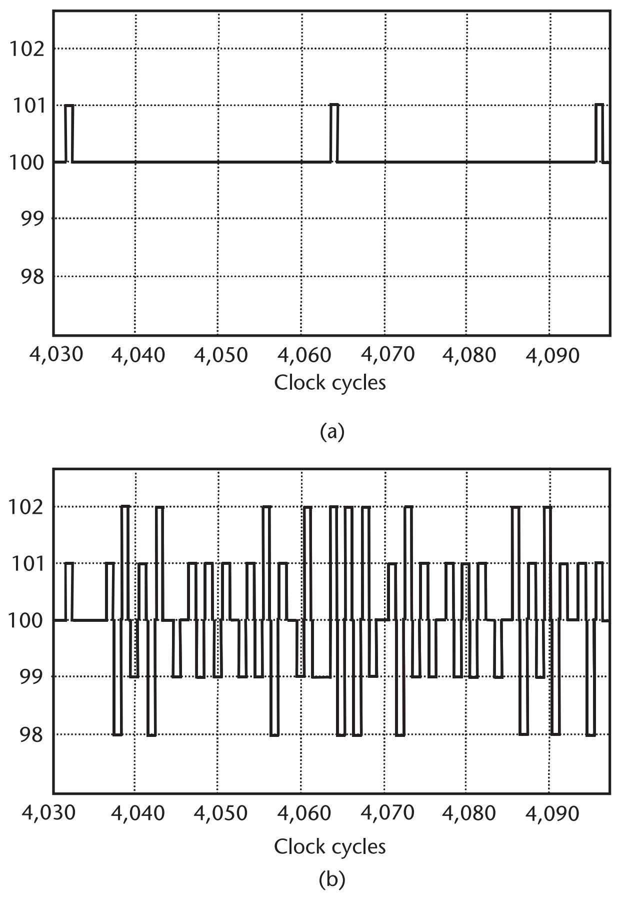


Figure 9.19 Instantaneous divisor ratio for (a) a fractional accumulator, and (b) a triple-loop ΣΔ accumulator with loop divisor $N = 100 + 1/32$.

frequencies, as shown in Figure 9.20(b). Obviously, the fractional spurs look more like frequency noise than discrete tones in the frequency spectrum after the ΣΔ noise shaping.

For a single-loop fractional-*N* accumulator ($m = 1$), the binary carry-out dithers the loop divider in the range of $I(z)$ to $I(z) + 1$ since C_1 is a 1-bit number that can have a value of either 0 or 1. For a dual-loop MASH ΣΔ accumulator ($m = 2$), the carry-out C_1 can have a value of either 0 or 1, and $C_2 - z^{-1}C_2$ can have values of -1, 0, or 1; thus, the sum will dither the loop divider in the range of $I(z) - 1$ to $I(z) + 2$. Similarly, for a triple-loop MASH ΣΔ accumulator ($m = 3$), the carry-out C_1, $C_2 - z^{-1}C_2$, and $C_3 - 2z^{-1}C_3 + z^{-2}C_3$ will dither the loop divider in the range of $I(z) - 3$ to $I(z) + 4$. Thus, the higher the ΣΔ modulator order, the wider the range over which the loop divider ratio is dithered. In general, an m-loop MASH ΣΔ modulator causes the loop divisor to be dithered in the range of $I(z) - 2^{m-1} + 1$ to $I(z) + 2^{m-1}$.

Having studied the noise-shaping characteristic of a multiloop ΣΔ modulator, the implementation for the architecture presented in Figure 9.18 will now be

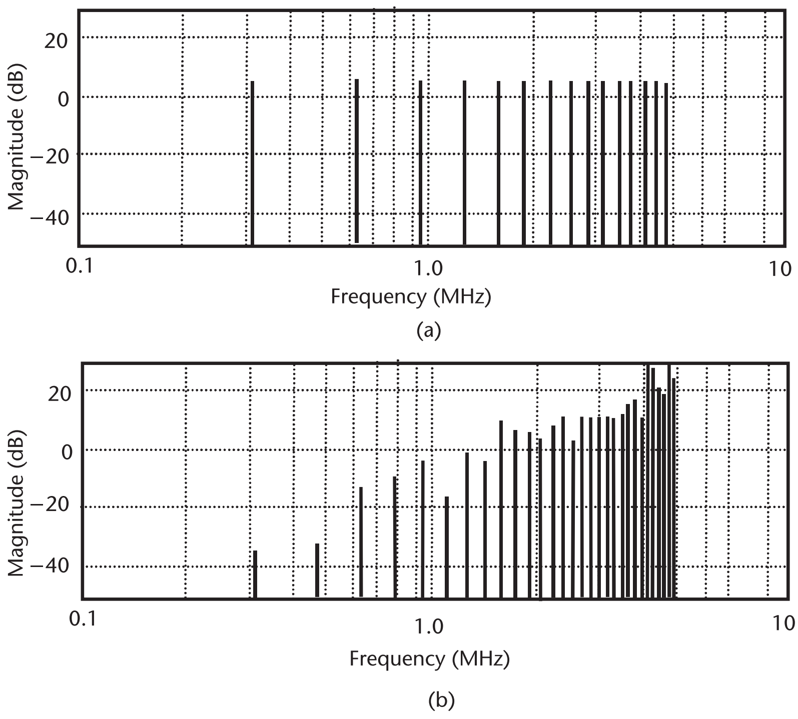


Figure 9.20 The output spectrum for (a) a fractional accumulator, and (b) a triple-loop ΣΔ accumulator for $N(z) = 100 + 1/32$ and the comparison frequency $f_r = 10$ MHz.

examined in more detail. First, the function $1/(1 - z^{-1})$ is an integration function and can be implemented with a single, readily available accumulator circuit. Second, the function of the quantizer will now be examined. Suppose the accumulator has an n-bit input word. If the carry-out bit needs to be preserved, the output of the accumulator is an $n + 1$ bit word A_i, where A_i is the output of the ith accumulator shown in Figure 9.18. The quantizer truncates the $(n + 1)$-bit word A_i by choosing its MSB as the accumulator output. The MSB is simply the carry-out bit C_i of the accumulator. This process introduces a quantization error of $E_{qi} = C_i - A_i$, which is fed into the next accumulator for noise shaping. When the accumulator does not overflow ($C_i = 0$), the 1-bit loop feedback does not have any impact on the accumulator performance. When the accumulator overflows ($C_i = 1$), the 1-bit loop feedback subtracts C_i from the accumulator input .$F(z)$. Note that, for the purpose of the subtraction, C_i takes on a value of {1'b1, n{1'b0}}, where the Verilog concatenation notation represents an $(n + 1)$-bit binary word with the MSB = 1 and the rest of the bits as zeros. In other words, the 1-bit C_i is multiplied by the module number of the accumulator before being input to the adder because it has the weight of the $(n + 1)$th bit. This process is simply the overflow of an n-bit accumulator. The accumulator starts from the residue value $A_i - C_i$ after an overflow occurs. The operation of other loops is equivalent to that of the first loop, except their inputs are the truncated quantization errors of the previous loop. Consequently, the mathematical model of a multiloop ΣΔ modulator shown in Figure 9.18 can be simply implemented by using the topology illustrated in

Figure 9.21. Following a trivial derivation, the transfer function of individual loops shown in Figure 9.21 can be obtained as

$$
\begin{aligned}
C_1(z) &= A_1(z) + E_{q1}(z) = [.F(z) - z^{-1}E_{q1}(z)] + E_{q1}(z) \qquad (9.44)\\
&= .F(z) + (1 - z^{-1})E_{q1}(z)
\end{aligned}
$$

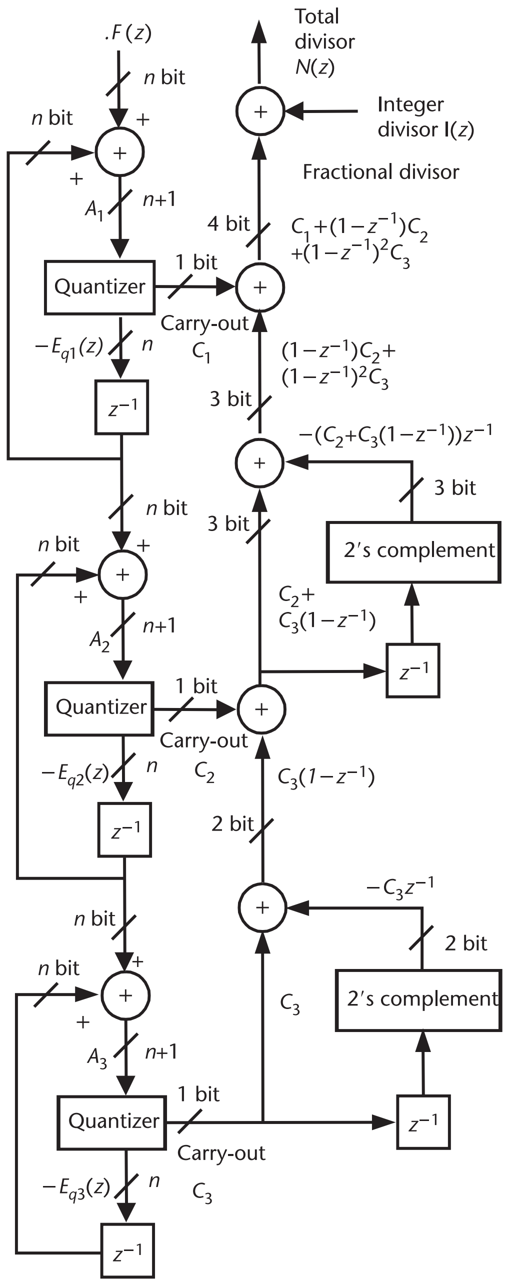


Figure 9.21 An alternative topology of the MASH 1-1-1 ΣΔ modulator using three accumulators.

$$C_2(z) = -E_{q1}(z) + (1 - z^{-1})E_{q2}(z) \tag{9.45}$$

$$C_3(z) = -E_{q2}(z) + (1 - z^{-1})E_{q3}(z) \tag{9.46}$$

Therefore, the overall output of the MASH 1-1-1 structure becomes

$$\begin{aligned} N(z) &= I(z) + C_1(z) + (1 - z^{-1})C_2(z) + (1 - z^{-1})^2 C_3(z) \\ &= I(z) + .F(z) + (1 - z^{-1})^3 E_{q3}(z) \end{aligned} \tag{9.47}$$

which is equivalent to (9.37). Note that the function $1 - z^{-1}$ is implemented as a differentiator circuit in Figure 9.21. Thus, the entire ΣΔ modulator includes only adders and delay units z^{-1}, which can be implemented with clocked flip-flops. The speed bottleneck of the multiloop ΣΔ modulator is the n-bit adder with carry-out bit. The postprocessing of the carry-out bits can be implemented using pipelined feedforward architectures. The MASH structure is absolutely stable and is very suitable for high-speed applications.

To understand the number of bits assigned to each adder, a review of the arithmetic operation starting from the 2-bit adder at the bottom of Figure 9.21 is needed. The carry-out bits C_1, C_2, and C_3 from the three accumulators can only be either 0 or 1. The 2-bit adder output $C_3(1 - z^{-1})$ can thus be –1, 0, or 1. Adding C_2 results in a 3-bit term $C_2 + C_3(1 - z^{-1})$, ranging from –1, 0, 1, or 2, which can be represented by a 3-bit two's complement word. Letting $A = C_2 + C_3(1 - z^{-1})$ and $B = [C_2 + C_3(1 - z^{-1})]z^{-1}$, the 3-bit subtraction of $(A - B)$ leads to a 3-bit two's complement word ranging from –3 to +3. As an example, if $A = -1$ = 3'b111 and $B = +2$ = 3'b010, $-B = -2$ = 3'b110, $A - B$ = 3'b111 + 3'b110 = 3'b101 = –3. Adding C_1 with $A - B$, we have a 4-bit word ranging from –3 to +4. Analyzing the modulator output in Figure 9.21, it is evident that the third-order MASH ΣΔ modulator has a 4-bit output with a maximum value of 4 and a minimum value of –3. Hence, the modulator dithers the loop divisor in the range from $I(z) - 3$ to $I(z) + 4$.

In addition to the MASH 1-1-1 topology shown in Figure 9.18, the highpass noise transfer function of $(1 - z^{-1})^m$ can also be implemented using a combination of loops with different orders. For a third-order ΣΔ modulator, a MASH 1-2 topology shown in Figure 9.22 consists of a first-order loop and a second-order loop. The advantage of a MASH 1-2 ΣΔ modulator is its simplified topology with reduced hardware and power consumption.

The MASH 1-1-1 structure is a widely used topology due to its stability, high-order in-band noise-shaping characteristic, and easy implementation. As shown, a ΣΔ modulator dithers the loop division ratio around its average value. Instantaneously, there are always small phase errors for a ΣΔ modulated PLL. However, the average phase error ought to be zero in order for the loop to lock to the desired frequency. Unfortunately, for a multiloop MASH ΣΔ modulator, the higher the order of the modulator, the larger the phase error it causes. To improve the phase error distribution without degrading the noise-shaping slope, alternative topologies need to be investigated.

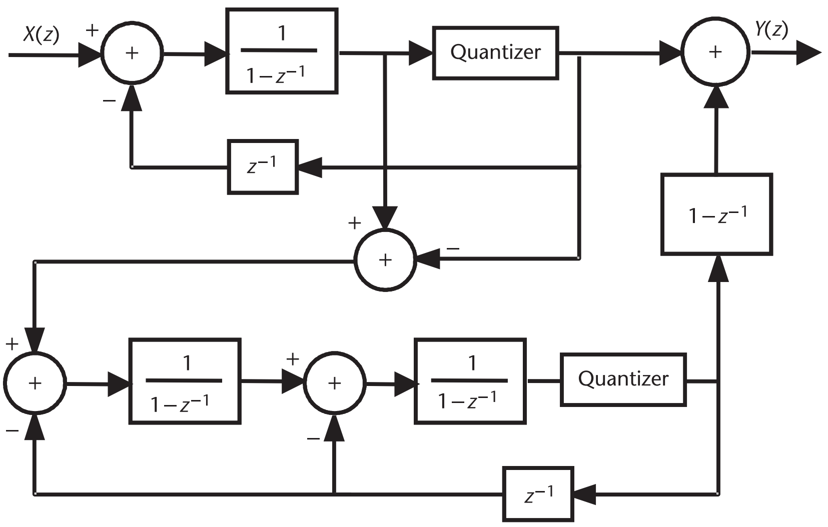


Figure 9.22 A third-order MASH 1-2 ΣΔ modulator.

9.3.3 Single-Stage ΣΔ Modulators with Multiple Feedback Paths

The noise transfer function of $(1 - z^{-1})^m$ can also be implemented using single-stage and multiple feedback paths, as illustrated in Figure 9.23. While a MASH 1-1-1 is absolutely stable, a single-stage feedback ΣΔ modulator is conditionally stable with reduced input ranges due to the feedback loops. A single-stage feedback ΣΔ modulator can output either a single bit or multibits, while a MASH 1-1-1 or a MASH 1-2 ΣΔ modulator can only output multibits. Using a single-stage ΣΔ modulator, its number of output bits can be chosen based on the desired output quantization noise level and the range of the loop divisor variations.

For the single-stage feedback ΣΔ modulator shown in Figure 9.23, the *m* accumulators have the following transfer functions:

$$H_i = \frac{1}{1 - z^{-1}} \quad i = 1, 2, \ldots, m - 1 \quad \text{and} \quad H_m = \frac{z^{-1}}{1 - z^{-1}} \tag{9.48}$$

The single-stage feedback ΣΔ modulator output can be found to be

$$Y(z) = H_x(z)X(z) + H_e(z)E(z) \tag{9.49}$$

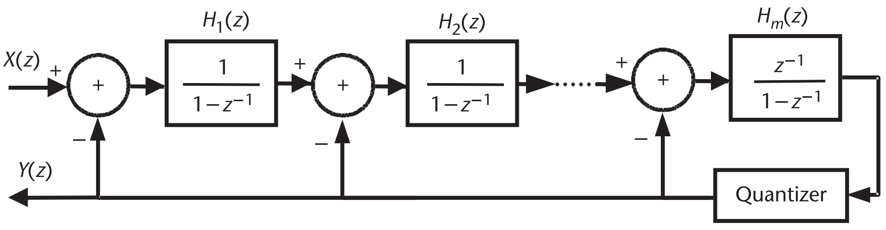


Figure 9.23 An *m*th-order, single-stage, feedback ΣΔ modulator.

where the signal transfer function is given by

$$H_x(z) = \frac{\left(\frac{1}{1-z^{-1}}\right)^{m-1}\left(\frac{z^{-1}}{1-z^{-1}}\right)}{1+\left(\frac{z^{-1}}{1-z^{-1}}\right)\sum_{i=0}^{m-1}\left(\frac{1}{1-z^{-1}}\right)^{i}} = z^{-1} \tag{9.50}$$

and the noise transfer function is given by

$$H_e(z) = \frac{1}{1+\left(\frac{z^{-1}}{1-z^{-1}}\right)\sum_{i=0}^{m-1}\left(\frac{1}{1-z^{-1}}\right)^{i}} = (1-z^{-1})^m \tag{9.51}$$

It is evident that the signal $X(z)$ experiences only a transport delay z^{-1}, while the quantization noise is suppressed by a highpass noise transfer function of $(1-z^{-1})^m$.

9.3.4 Single-Stage ΣΔ Modulators with a Single Feedback Path

In addition to the single-stage ΣΔ modulator using multiple feedback paths given in Figure 9.23, an alternative single-stage ΣΔ modulator is proposed in [7]. Conceptually, by inserting a block with a transfer function of $H(z) = 1 - H_e(z)$ in an accumulator as shown in Figure 9.24, the accumulator output becomes

$$Y(z) = X(z) + A(z)H(z) - A(z) = X(z) - E(z)H_e(z) \tag{9.52}$$

where $Y(z)$ is the most significant p bits of the adder output $B(z)$, and $A(z)$ is the remaining $(n+1-p)$ bits of the adder output $B(z)$. It is evident that the input signal $X(z)$ is not affected by the modulator, while the quantization noise $E(z)$, which is a truncated word $A(z)$, is filtered by the noise transfer function of $H_e(z)$. If the noise transfer function $H_e(z)$ is the highpass transfer function $(1-z^{-1})^m$,

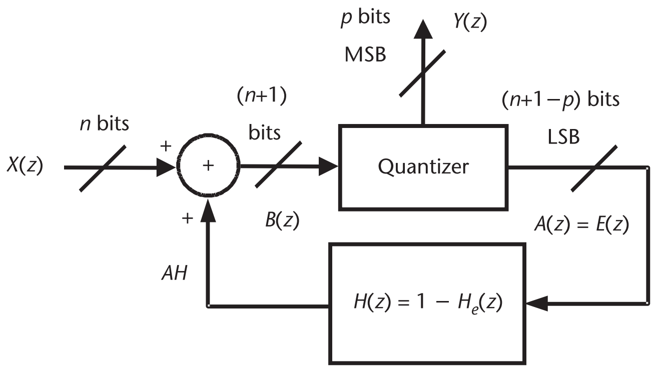


Figure 9.24 A conceptual drawing of the single-stage ΣΔ modulator.

and the feedback transfer function is therefore $H(z) = 1 - (1 - z^{-1})^m$, the single-stage modulator is equivalent to a MASH modulator with $Y(z) = X(z) - E(z)(1 - z^{-1})^m$. Notice that the ΣΔ modulator degenerates to a standard fractional accumulator when $m = 1$ and $p = 1$ (1-bit carry-out). Thus, for $m = 1$, this structure is equivalent to a standard fractional accumulator, which does not have noise shaping. If an input frequency word $X(z)$ has n bits, $B(z)$ should have $(n + 1)$ bits to include the carry-out, and $A(z)H(z)$ cannot exceed n bits. The modulator output $Y(z)$ can be of any number of bits, offering flexibility in choosing the number of output bits. However, the maximum number of bits for $A(z)H(z)$ should be carefully calculated to prevent overflow of the adder.

The conceptual single-stage ΣΔ modulator shown in Figure 9.24 with $H(z) = 1 - (1 - z^{-1})^m$ can be implemented in a pipelined topology, where data flow is optimized for speed. For $m = 2$, $H(z) = 1 - (1 - z^{-1})^2 = 2z^{-1} - z^{-2}$. The implementation of the second-order modulator is presented in Figure 9.25, where the subtraction is implemented using two's complement format for the z^{-2} term, and multiplication by two is implemented using a left shift operation. If $n + 2 - p < n$, sign extension is performed by extending the MSB of the $(n + 2 - p)$-bit word to obtain an n-bit word. As can be seen, a condition of $n + 2 - p \leq n$ needs to hold to prevent the first adder with an $(n + 1)$-bit output from losing the overflow bits. Hence, the minimum number of output bits of the second-order modulator is two. In general, the number of output bits for the given single-stage ΣΔ accumulator should be equal to or larger than the order of the modulator m.

For $m = 3$, $H(z) = 1 - (1 - z^{-1})^3 = z^{-1}(3 - 3z^{-1} + z^{-2})$. The implementation of the third-order, single-stage ΣΔ modulator is given in Figure 9.26. Again, the subtraction is implemented using two's complement format. Multiplication by three is implemented using a left shift operation (×2), followed by addition, given mathematically by $3z^{-1} = 2z^{-1} + z^{-1}$. Figures 9.27 and 9.28 show the implementation of the fourth-order and fifth-order single-stage ΣΔ modulators, respectively. They are somewhat more complicated compared to the second- and third-order modulators. The fourth-order, single-stage ΣΔ modulator uses a transfer function of $H(z) = 1 - (1 - z^{-1})^4 = 4z^{-1} - 6z^{-2} + 4z^{-3} - z^{-4}$, while the fifth-order single-stage ΣΔ modulator has a transfer function of $H(z) = 1 - (1 - z^{-1})^5 = 5z^{-1} -$

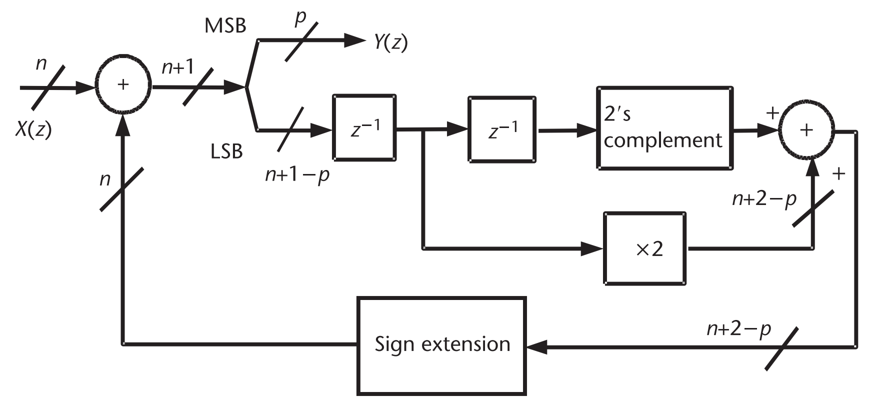


Figure 9.25 A second-order, single-stage ΣΔ modulator.

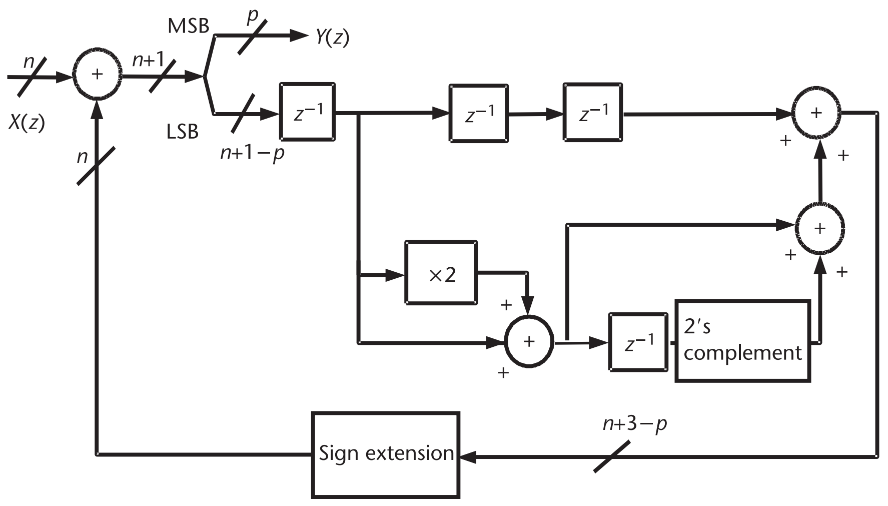


Figure 9.26 A third-order, single-stage ΣΔ modulator.

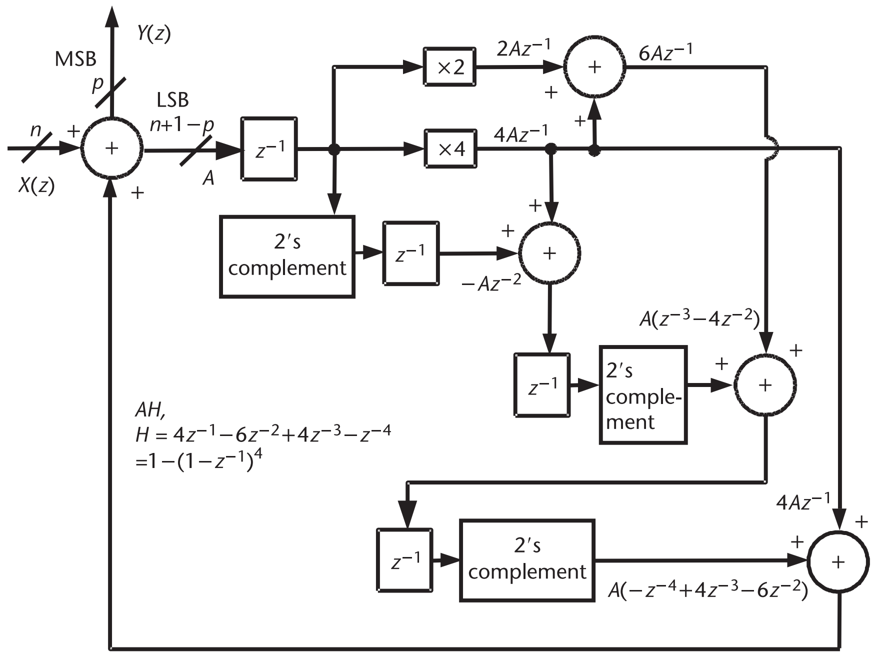


Figure 9.27 A fourth-order, single-stage ΣΔ modulator.

$10z^{-2} + 10z^{-3} - 5z^{-4} + z^{-5}$. To speed up the circuits, those transfer functions are implemented in a pipelined manner. To avoid using multipliers, which can be area and speed bottlenecks, the transfer function $H(z)$ is manipulated such that only shifting operations are involved. The speed of the single-stage ΣΔ modulator topology is limited by the delay times associated with the additions to calculate the transfer function $H(z)$. The higher the order of the ΣΔ modulator, the longer the delay.

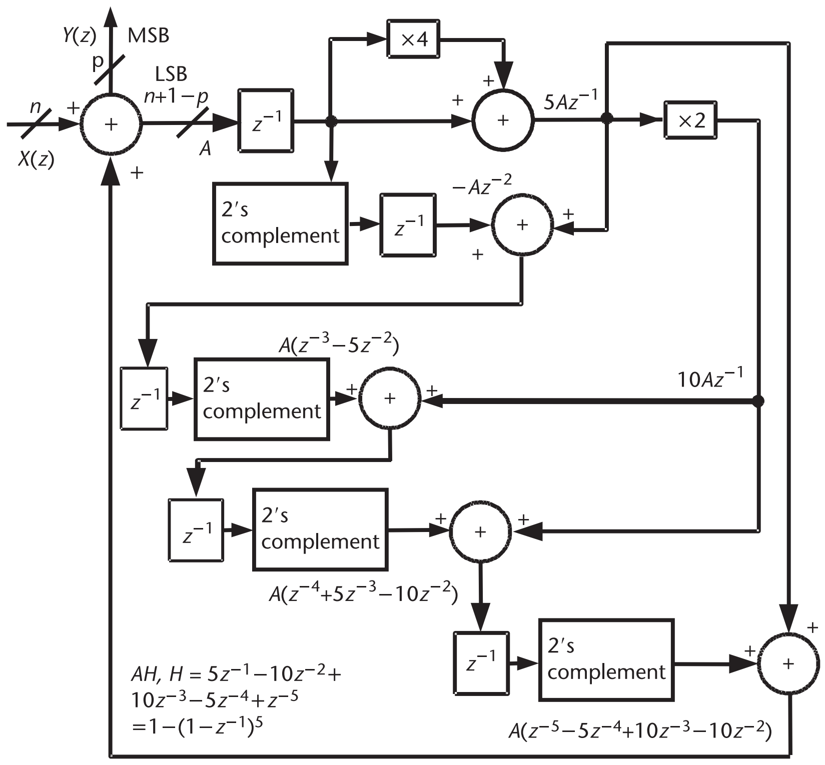

Y(z)
MSB
p
LSB
n+1−p
n
X(z)
A
z^{-1}
×4
5Az^{-1}
×2
2's complement
−Az^{-2}
A(z^{-3}−5z^{-2})
10Az^{-1}
A(z^{-4}+5z^{-3}−10z^{-2})
AH, H = 5z^{-1}−10z^{-2}+ 10z^{-3}−5z^{-4}+z^{-5} =1−(1−z^{-1})^5
A(z^{-5}−5z^{-4}+10z^{-3}−10z^{-2})

Figure 9.28 A fifth-order, single-stage ΣΔ modulator.

9.3.5 A Generic High-Order ΣΔ Modulator Topology

The above sections have studied multistage and single-stage ΣΔ modulators for fractional-*N* synthesis. All of the topologies studied so far have a common noise transfer function of

$$H_e(z) = (1 - z^{-1})^m = \left(\frac{z-1}{z}\right)^m \tag{9.53}$$

This highpass noise transfer function has m zeros at $z = 1$ and m poles at the origin of the z plane. It provides noise shaping to reject the in-band noise and spurs, yet the noise-shaping slope extends out of band to higher frequencies, which requires a high-order loop filter for attenuation. In the time domain, the MASH ΣΔ modulator tends to generate high-frequency bit patterns with widely spread output values, which makes the design of the phase detector and charge pump challenging. The widely spread output values lead to a wide spread in loop-division ratio around the desired value. This tends to increase the charge pump turn-on time, which increases the reference noise feedthrough and the crosstalk through the substrate. As a result, the PLL in-band phase noise is degraded. Therefore, a more efficient noise transfer function than the traditional MASH transfer function of $(1 - z^{-1})^m$ is often desirable. A possible step to modify the MASH transfer function would be adding poles at a higher frequency to reduce the high-frequency noise caused by the MASH topology. However, for every modification of the noise transfer function, an implementation with minimum hardware overhead should be considered. This section introduces a generic ΣΔ modulator architecture, which can be implemented with feedforward and feedback paths. First, it is necessary to review the high-order ΣΔ modulator presented in Figure 9.12, in which cascade multiloops are used to generate the mth-order MASH noise transfer function, with m zeros at $z = 1$ and m poles at $z = 0$. To design a noise transfer function with arbitrary zeros, feedforward paths with different gains can be added. To add arbitrary poles to the noise transfer function, either feedforward or feedback paths with different gains can be added.

An integrator or an accumulator can be implemented in two different ways, as shown in Figure 9.29. The integrator transfer function differs slightly, depending on the location of the delay elements. When multiple integrators with a transfer function of $1/(1 - z^{-1})$ are cascaded, there are no delay elements in the signal path. However, there will still be a propagation delay through each element. Therefore, their output signals will change asynchronously. This may limit the maximum clock speed, making this unsuitable for high-speed applications. On the other hand, when multiple integrators with the transfer function of $z^{-1}/(1 - z^{-1})$ are cascaded, there are clocked delays in the signal path. Hence, all adder inputs change synchronously, which benefits the circuit speed. However, it is often necessary to use different integrator topologies intelligently to achieve the desired transfer functions, as will be seen in a later example. Generic ΣΔ modulators using both integrator topologies will be discussed.

One variant of the ΣΔ topology is to place a delay element in the loop feedback path, as shown in Figure 9.17. This type of modulator originated in data-conversion applications. Adding a delay element z^{-1} in the loop feedback is not necessary in frequency synthesis. However, for a ΣΔ ADC, there will be a DAC in the loop feedback to convert the digital quantized data into an analog signal since the input to the ADC is an analog signal. Note that there is a delay element in the conceptual ΣΔ modulator drawn in Figure 9.17 to model the delay through a DAC. The four topologies of the generic ΣΔ modulators discussed below provide variants to allow the design of a wide range of noise transfer functions. The combination of those variants to form a desired transfer function is possible, as the next section shows.

9.3.5.1 Generic ΣΔ Modulator Topology with Integrators $1/(1 - z^{-1})$ and Without Delay in the Loop Feedback Path

First, consider the integrator without a delay element in the feedback path. A simple modification can be made to the first-order ΣΔ modulator given in Figure 9.10 by

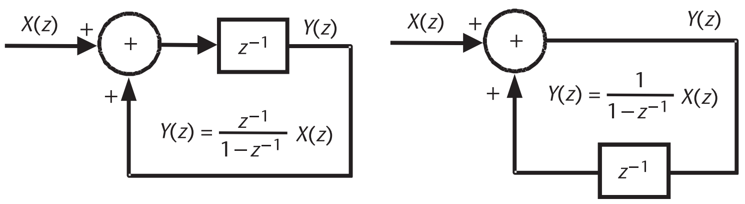


Figure 9.29 Integrator topologies with delay elements at different locations.

adding two feedforward coefficients, A_0 and A_1, and one feedback coefficient, B_1. Figure 9.30 shows the resulting architecture. The signals in the modulator are easily shown to be given by

$$\begin{aligned} Y(z) &= D(z) + E_q(z) \\ Y(z) &= C_0(z)A_0 + C_1(z)A_1 + E_q(z) = C_0(z)[A_0 + A_1(1 - z^{-1})^{-1}] + E_q(z) \\ C_0(z) &= X(z) - Y(z) + C_1(z)B_1 \end{aligned} \tag{9.54}$$

Consequently, the following transfer function for the modified first-order ΣΔ modulator can be derived as

$$\begin{aligned} Y(z) = {} & \frac{A_0(1 - z^{-1}) + A_1}{(1 - z^{-1}) - B_1 + [A_0(1 - z^{-1}) + A_1]} X(z) \\ & + \frac{(1 - z^{-1}) - B_1}{(1 - z^{-1}) - B_1 + [A_0(1 - z^{-1}) + A_1]} E_q(z) \end{aligned} \tag{9.55}$$

Adding feedforward and feedback coefficients provides the flexibility to place poles and zeros at a desired frequency for in-band noise shaping and for damping the high-frequency noise. This simple modification can be generalized to any order of the ΣΔ modulator with both feedforward and feedback paths, as shown in Figure 9.31 [8].

The resultant transfer function of the generic ΣΔ modulator can be found to be

$$Y(z) = H_X(z)X(z) + H_E(z)E_q(z) \tag{9.56}$$

where the signal transfer function is given by

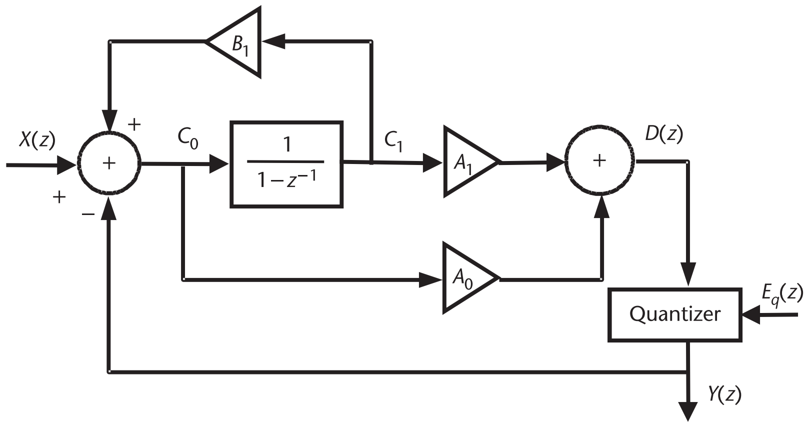


Figure 9.30 A first-order ΣΔ modulator with feedforward and feedback paths and an integrator $1/(1 - z^{-1})$.

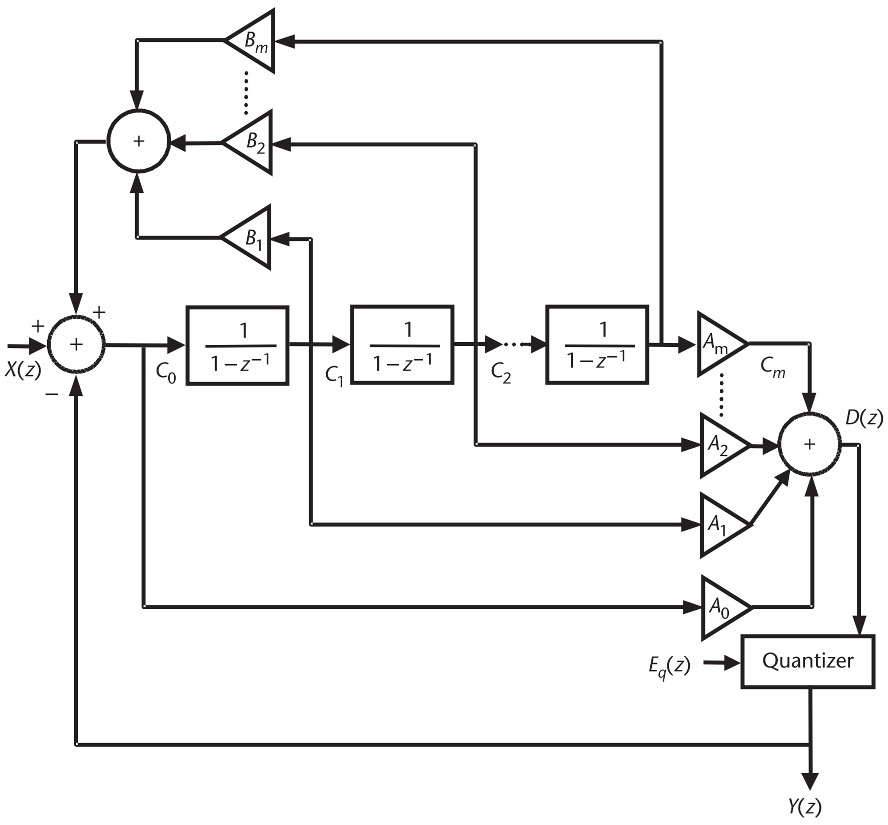


Figure 9.31 An *m*th-order ΣΔ modulator with feedforward and feedback paths.

$$H_X(z) = \frac{\sum_{i=0}^{m} A_i(1 - z^{-1})^{m-i}}{(1 - z^{-1})^m - \sum_{i=1}^{m} B_i(1 - z^{-1})^{m-i} + \sum_{i=0}^{m} A_i(1 - z^{-1})^{m-i}} \tag{9.57}$$

and the noise transfer function is given by

$$H_E(z) = \frac{(1 - z^{-1})^m - \sum_{i=1}^{m} B_i(1 - z^{-1})^{m-i}}{(1 - z^{-1})^m - \sum_{i=1}^{m} B_i(1 - z^{-1})^{m-i} + \sum_{i=0}^{m} A_i(1 - z^{-1})^{m-i}} \tag{9.58}$$

As discussed, the MASH ΣΔ modulator has good in-band noise-shaping characteristics, yet its noise needs to be further attenuated at high frequency. In the z domain, this translates into adding proper poles at high frequency and leaving the zeros, as discussed in earlier sections. Therefore, the feedforward technique is often more useful than using feedback. For an *m*th-order feedforward ΣΔ modulator with $B_i = 0$, the noise transfer function given in (9.58) becomes

$$H_E(z) = \frac{(1 - z^{-1})^m}{(1 - z^{-1})^m + \sum_{i=0}^{m} A_i (1 - z^{-1})^{m-i}} \tag{9.59}$$

For frequencies much smaller than sampling frequency, $1 - z^{-1} = 1 - e^{j\Omega} = 1 - \cos\Omega + j\sin\Omega \approx j\Omega$, where $\Omega = 2\pi f/f_s$. Ignoring the higher-order terms, the simplified expression for the output is

$$Y(z) \approx X(z) + E_q(z)\frac{(j\Omega)^m}{A_m} \tag{9.60}$$

When the order of the modulator and the oversampling ratio are sufficiently large, we have $|j\Omega|^m \ll 1$. Thus, the quantization noise $E_q(z)$ is greatly attenuated, which demonstrates the accurate tracking of the modulator at low frequencies. Recall that the VCO output signal tracks the PLL input only at low frequencies. In the time domain, this means that the average loop division ratio is accurately tracking the desired value, even though the instantaneous division ratio (seen as higher-frequency spectral components) is never the correct value for ΣΔ fractional-*N* synthesis.

9.3.5.2 Generic ΣΔ Modulator Topology with Integrators $z^{-1}/(1 - z^{-1})$ and Without Delay in the Loop Feedback Path

Second, consider the integrator with a delay element in the signal path. The modified first-order ΣΔ modulator with feedback and feedforward paths is illustrated in Figure 9.32.

The signals in the above structure can be found as follows:

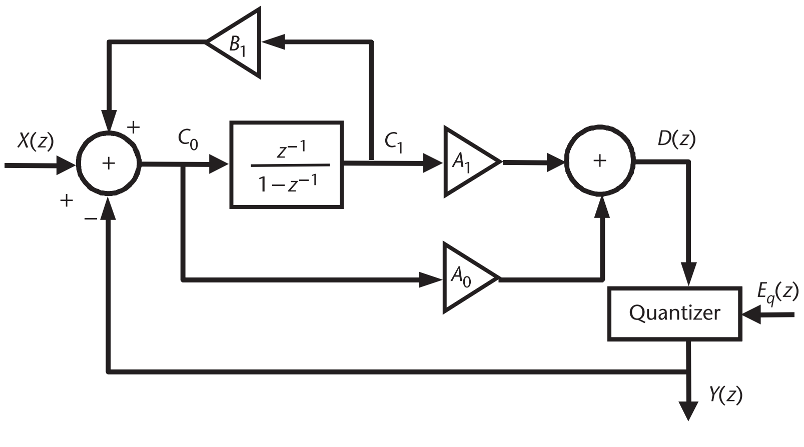


Figure 9.32 A first-order ΣΔ modulator with feedforward and feedback paths and an integrator $z^{-1}/(1 - z^{-1})$.

$$Y(z) = D(z) + E_q(z) \tag{9.61}$$

$$Y(z) = C_0(z)A_0 + C_1(z)A_1 + E_q(z) = C_0(z)\left(A_0 + A_1\frac{z^{-1}}{1 - z^{-1}}\right) + E_q(z)$$

Considering

$$C_0(z) = X(z) - Y(z) + C_1(z)B_1 \tag{9.62}$$

$$C_0(z) = [X(z) - Y(z)]\left(\frac{1 - z^{-1}}{1 - z^{-1} - B_1 z^{-1}}\right)$$

the transfer function for the modified first-order ΣΔ modulator can be derived as

$$Y(z) = \frac{A_0(z-1) + A_1}{z - 1 - B_1 + A_0(z-1) + A_1}X(z) + \frac{z - 1 - B_1}{z - 1 - B_1 + A_0(z-1) + A_1}E_q(z) \tag{9.63}$$

The above equation degenerates to $Y(z) = X(z)z^{-1} + (1 - z^{-1})E_q(z)$ when $A_0 = 0$, $A_1 = 1$, and $B_1 = 0$. In other words, the architecture without feedforward and feedback paths has the same transfer function as that of the MASH structure. If all the integrators shown in Figure 9.31 with the transfer function of $1/(1 - z^{-1})$ are replaced with integrators with a transfer function of $z^{-1}/(1 - z^{-1})$, the generic ΣΔ modulator topology ends up with a signal transfer function of

$$H_X(z) = \frac{\sum_{i=0}^{m} A_i(z-1)^{m-i}}{(z-1)^m - \sum_{i=1}^{m} B_i(z-1)^{m-i} + \sum_{i=0}^{m} A_i(z-1)^{m-i}} \tag{9.64}$$

and a noise transfer function of

$$H_E(z) = \frac{(z-1)^m - \sum_{i=1}^{m} B_i(z-1)^{m-i}}{(z-1)^m - \sum_{i=1}^{m} B_i(z-1)^{m-i} + \sum_{i=0}^{m} A_i(z-1)^{m-i}} \tag{9.65}$$

It is easy to prove that (9.63) is a special case of the transfer functions given in (9.64) and (9.65) for the first-order loop.

9.3.5.3 Generic ΣΔ Modulator Topology with Integrators $z^{-1}/(1 - z^{-1})$ and with a Delay in the Loop Feedback Path

Next, consider a slightly different ΣΔ topology, in which a delay element is added in the loop feedback path. Starting with the conceptual ΣΔ modulator drawn in

Figure 9.17 and adding feedback and feedforward paths results in the modified ΣΔ modulator shown Figure 9.33. The transfer function for the modified first-order ΣΔ modulator is given by

$$Y(z) = \frac{A_0(z-1)+A_1}{z[(z-1)-B_1]+[A_0(z-1)+A_1]}X(z) + \frac{(z-1)-B_1}{z[(z-1)-B_1]+[A_0(z-1)+A_1]}E_q(z) \tag{9.66}$$

Comparing the above transfer function to (9.63), it can be seen that the first part of the denominator $(z-1-B_1)$ is multiplied by variable z due to the delay in the loop feedback.

In general, for the topology of the mth-order ΣΔ modulator shown in Figure 9.34 with integrators of $z^{-1}/(1-z^{-1})$, a delay element in the loop feedback path introduces a z term in the denominator, and the generic ΣΔ modulator transfer functions become

$$H_X(z) = \frac{\sum_{i=0}^{m} A_i(z-1)^{m-i}}{z\left[(z-1)^m - \sum_{i=1}^{m} B_i(z-1)^{m-i}\right] + \sum_{i=0}^{m} A_i(z-1)^{m-i}} \tag{9.67}$$

and

$$H_E(z) = \frac{(z-1)^m - \sum_{i=1}^{m} B_i(z-1)^{m-i}}{z\left[(z-1)^m - \sum_{i=1}^{m} B_i(z-1)^{m-i}\right] + \sum_{i=0}^{m} A_i(z-1)^{m-i}} \tag{9.68}$$

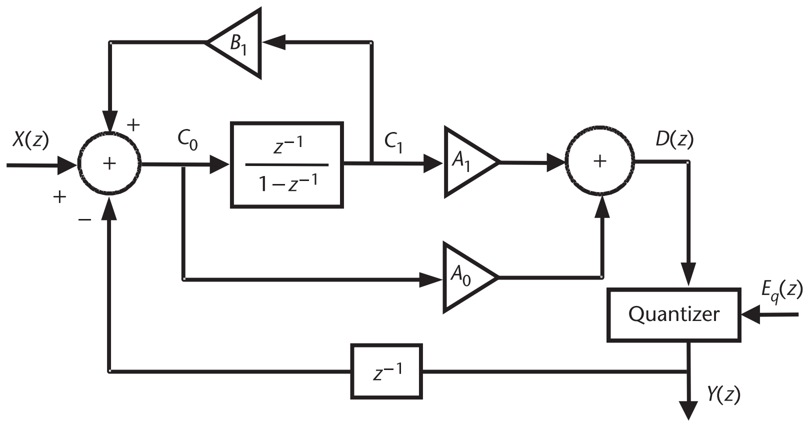


Figure 9.33 A first-order ΣΔ modulator with feedforward (A_0, A_1) and feedback (B_1) paths, an integrator $z^{-1}/(1-z^{-1})$, and a delay z^{-1} in the loop feedback.

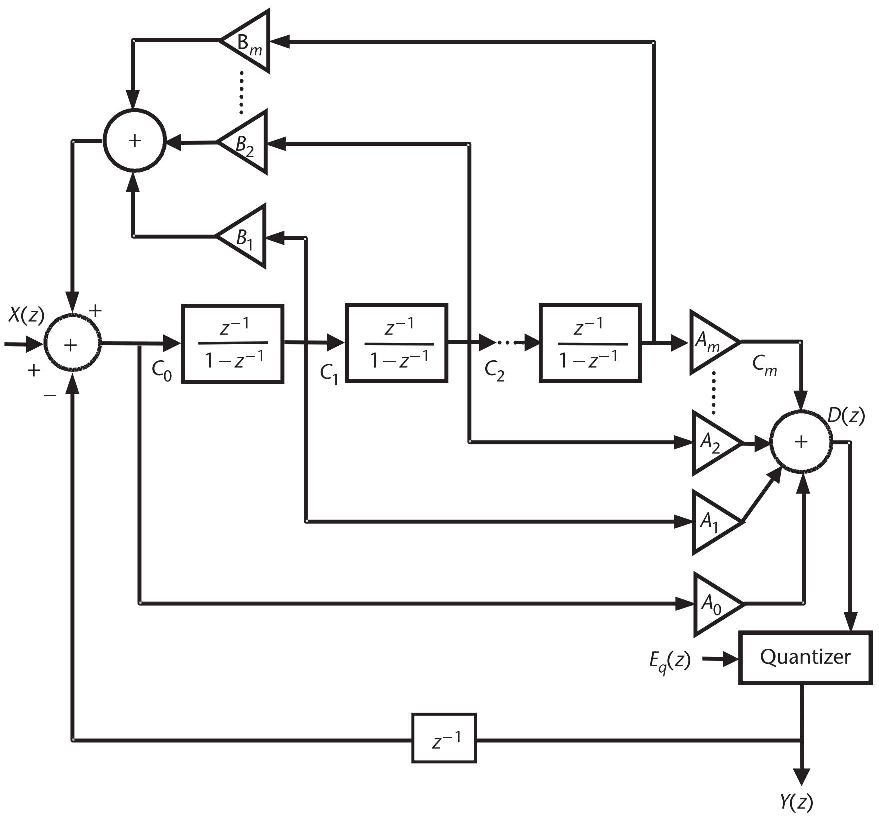


Figure 9.34 An alternative *m*th-order ΣΔ modulator with feedforward and feedback paths.

9.3.5.4 Generic ΣΔ Modulator Topology with Integrators $1/(1 - z^{-1})$ and with a Delay in the Loop Feedback Path

Finally, for the topology of the *m*th-order ΣΔ modulator with integrators of $1/(1 - z^{-1})$, a delay element in the loop feedback path introduces a z^{-1} term in the denominator, and the generic ΣΔ modulator transfer functions given in (9.57) and (9.58) become

$$H_X(z) = \frac{\sum_{i=0}^{m} A_i(1 - z^{-1})^{m-i}}{(1 - z^{-1})^m - \sum_{i=1}^{m} B_i(1 - z^{-1})^{m-i} + z^{-1}\sum_{i=0}^{m} A_i(1 - z^{-1})^{m-i}} \tag{9.69}$$

with the noise transfer function as

$$H_E(z) = \frac{(1 - z^{-1})^m - \sum_{i=1}^{m} B_i(1 - z^{-1})^{m-i}}{(1 - z^{-1})^m - \sum_{i=1}^{m} B_i(1 - z^{-1})^{m-i} + z^{-1}\sum_{i=0}^{m} A_i(1 - z^{-1})^{m-i}} \tag{9.70}$$

9.3.6 Modified ΣΔ Modulator with Improved High-Frequency Response

The above developed generic ΣΔ modulator architectures can be used to modify the MASH ΣΔ structure for improved high-frequency response for fractional-*N* synthesis. This section gives examples of the generic ΣΔ modulator architectures. The selection of different coefficients and their impact on noise shaping for a single-stage multiple feedforward ΣΔ modulator will be investigated. As discussed previously, the MASH ΣΔ modulator has substantial high-frequency noise, which results in intensive modulus switching over a wide range. The goal of modifying the MASH noise transfer function is to attenuate its high-frequency noise by adding poles at high frequency.

Previously, a 1-bit quantizer was discussed as a building block of a ΣΔ modulator. However, the generic topology shown in Figure 9.31 can make use of a single-bit or multibit quantizer. For a multibit quantizer, the quantization power is given by $\Delta^2/12$, where $\Delta = \Delta N/(2^B - 1)$, where ΔN denotes the modulus range, and B is the number of significant ΣΔ output bits. More output bits reduce the quantization noise of a ΣΔ modulator, yet they cause larger variation of the loop division ratio and sometimes even require a larger MMD. In the MASH ΣΔ modulator, each cascaded loop provides a carry-out bit, which is further processed following the coefficients of a Pascal's triangle. Thus, the MASH ΣΔ modulator has to have a multibit output with the number of bits determined by the order of the modulator. This lack of flexibility for the MASH ΣΔ modulator is one of its disadvantages.

A MASH ΣΔ modulator is absolutely stable. However, for a ΣΔ modulator with feedforward or feedback paths, stability requires that all the discrete poles be placed inside the unit circle in the z domain. Thus, the feedforward or feedback coefficients should be accurately controlled. Although a coefficient can be modulated to a digital word by using a digital multiplier, such an approach requires too much hardware and also limits the modulator speed. The coefficients in a ΣΔ modulator are normally implemented using shifting and simple addition operations, which limits the choice of coefficients, as will be seen in later sections.

ΣΔ modulators are nonlinear systems, and their stability analysis is different from linear system analysis. Instability occurs when the input amplitude or the frequency of the ΣΔ modulator exceeds a certain value, which depends on the modulator structure. Under unstable conditions, low-frequency signal swing between the minimum and maximum amplitude occurs at the quantizer input. As a result, the quantizer output is saturated (overloaded), and the ΣΔ modulator can no longer track the input signal. For a single-bit quantizer, the saturated quantizer output corresponds to long sequences of ones followed by long sequences of zeros, also called limit cycles. It is very difficult for the modulator to get out of the saturated state; hence, the ΣΔ modulator becomes unstable.

9.3.6.1 Single-Stage Multiple Feedforward ΣΔ Modulator (SSMF-I)

A single-stage, multiple-feedforward ΣΔ modulator, the SSMF-I, with single-bit output was proposed in [6] and is illustrated in Figure 9.35. The single-bit quantizer can be implemented using the carry-out of the adder; however, this results in high quantization noise. Another disadvantage of the architecture is that the range over which the modulator is stable is reduced; hence, the tuning range of the modulator

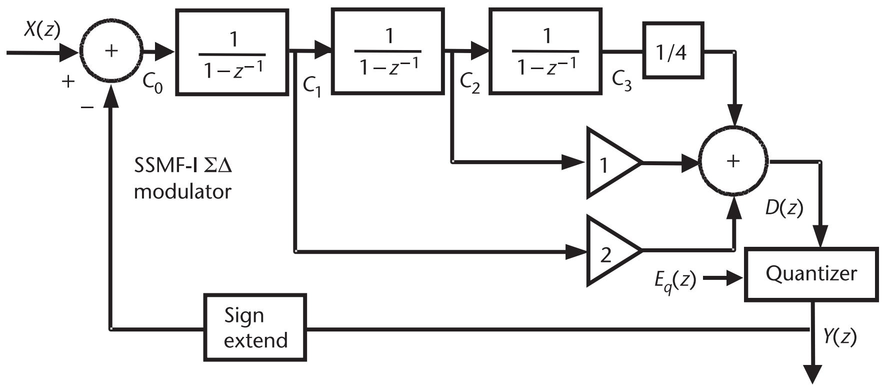


Figure 9.35 A third-order, single-stage, multiple-feedforward ΣΔ modulator (SSMF-I).

is reduced. Although the feedforward coefficients are all powers of two that can be implemented with shifting operations, the hardware complexity of the modulator is higher than that of the MASH modulator. The modulator does not have a feedback path; that is, all the coefficients B_i in (9.58) vanish. The coefficients for feedforward paths are $C_0 = 0$, $C_1 = 2$, $C_2 = 1$, and $C_3 = 1/4$. Based on (9.59), it is straightforward to find the noise transfer function of the SSMF-I ΣΔ modulator as follows:

$$H_{E1}(z) = \frac{(1-z^{-1})^3}{(1-z^{-1})^3 + [2(1-z^{-1})^2 + (1-z^{-1}) + 1/4]} = \frac{(1-z^{-1})^3}{4.25 - 8z^{-1} + 5z^{-2} - z^{-3}} \tag{9.71}$$

Due to the single-bit quantizer in the above third-order SSMF-I ΣΔ modulator, the instantaneous division ratio can only dither in a narrow range, which improves the PLL in-band noise since the charge pump turn-on time is reduced. In comparison, a MASH 1-1-1 ΣΔ modulator dithers the loop divisor over a wider range from $I(z) - 3$ to $I(z) + 4$. Unlike the MASH 1-1-1 ΣΔ modulator, the SSMF-I ΣΔ modulator presents attenuated high-frequency noise due to the additional poles. However, the single-bit quantizer causes slightly higher in-band phase noise. It is thus desirable to increase the number of quantizer bits, as the next example shows, which will reduce the in-band noise.

9.6.3.2 Single-Stage Multiple Feedforward ΣΔ Modulator (SSMF-II)

Another example of the modified ΣΔ modulator of this kind, SSMF-II, was presented in [9] and is illustrated in Figure 9.36. The modified ΣΔ modulator utilizes two additional feedforward paths for the first and second accumulator outputs. All the coefficients in the feedforward and feedback paths are powers of two, which can be implemented with shifting operations. A 5-bit quantizer is employed to achieve low quantization noise. Furthermore, dithering is used to introduce sufficient randomization in the modulator. Integrators with transfer functions of both $z^{-1}/(1-z^{-1})$ and $1/(1-z^{-1})$ are employed in order to obtain the following desired noise transfer function as

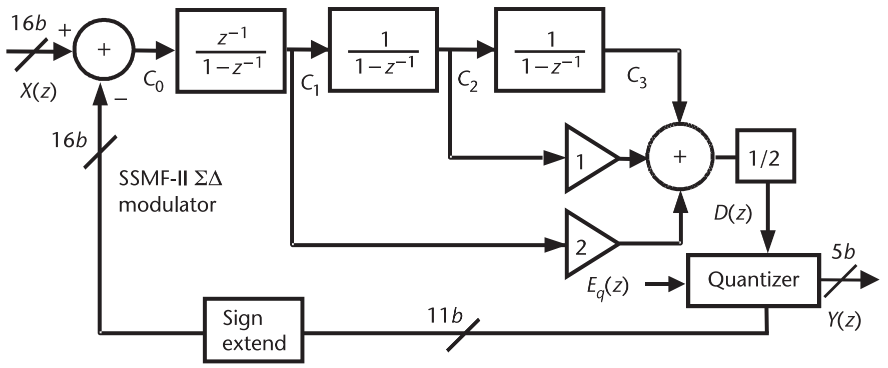


Figure 9.36 A third-order, single-stage, multiple-feedforward ΣΔ modulator (SSMF-II).

$$H_{E2}(z) = \frac{(1 - z^{-1})^3}{1 - z^{-1} + z^{-2}/2} \tag{9.72}$$

The above noise transfer function of the SSMF-II ΣΔ modulator can also be implemented in another configuration, as proposed in [10] and illustrated in Figure 9.37, where three output bits are used. The peak of the quantization noise is flattened by introducing an extra pole into the digital modulator. This approach helps to meet phase noise specifications at high-frequency offsets while still using a high-order modulator. Again, dithering is used to introduce sufficient randomization into the modulator. The disadvantages to this architecture are reduced tuning range and increased complexity compared to a MASH ΣΔ modulator.

The noise transfer function of the modulator topology presented in Figure 9.37 can be found from (9.65) and is given by

$$H_{E3}(z) = \frac{(z-1)^3}{(z-1)^3 + [2(z-1)^2 + 1.5(z-1) + 0.5]} = \frac{(1 - z^{-1})^3}{1 - z^{-1} + 0.5z^{-2}} \tag{9.73}$$

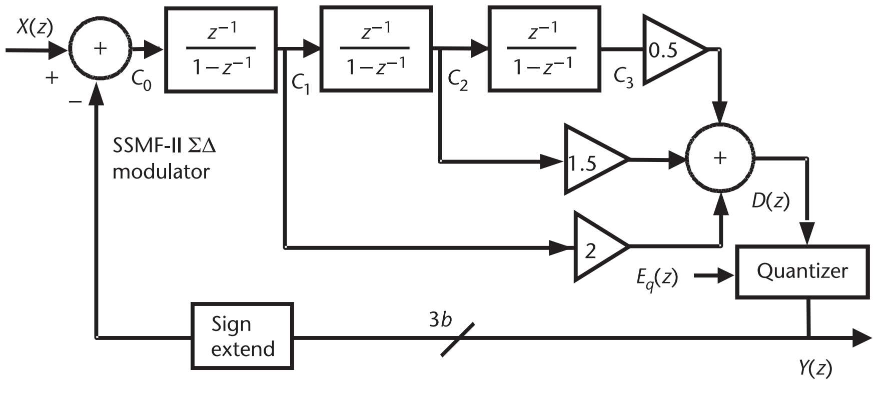


Figure 9.37 Alternate topology of the third-order, single-stage, multiple-feedforward ΣΔ modulator (SSMF-II).

which is the same noise transfer function as that given in (9.72). It can be shown that the modified ΣΔ modulator adds two low-Q Butterworth poles, in addition to the three zeros at $z = 1$ found in the standard MASH 1-1-1 structure. Figure 9.38 shows the pole-zero plot for the SSMF-I and SSMF-II ΣΔ modulators with noise transfer functions given in (9.71) and (9.72). All the poles for both the SSMF-I and SSMF-II ΣΔ modulators are located inside the unit circle. For the single-stage ΣΔ modulators described in Section 9.3.6, the condition for absolute stability requires that all the poles and zeros be placed inside the unit circle. In addition, the stable input range depends on the modulator topology, the type of input, and the number of input and quantizer bits. It sometimes takes certain long input sequences for instability in the form of limit cycles to be excited. Hence, extensive simulations need to be done using different types and lengths of input sequences to determine the input bounds for stability. In contrast to the high-order, single-stage loop modulators, the multiloop ΣΔ modulator described in Section 9.3.2 is absolutely stable.

The magnitudes of the noise transfer functions for SSMF-I and SSMF-II ΣΔ modulators are plotted with comparison to the MASH noise transfer function in Figure 9.39. The plot is also zoomed in to compare the in-band noise-shaping effect of the three modulators. It can be seen that the MASH noise transfer function has the best in-band noise filtering. For the frequency band $f < 0.06f_s = 0.06f_{\text{ref}}$, the SSMF-II modulator demonstrates better noise shaping than the SSMF-I modulator. As a rule of thumb, the loop bandwidth for an integer-N synthesizer PLL is chosen near $0.1f_{\text{ref}}$. The choice of loop bandwidth to minimize quantization noise due to the use of ΣΔ modulators is discussed in Section 9.3.11. At high frequencies (beyond the loop bandwidth), the MASH noise transfer function is not attenuated,

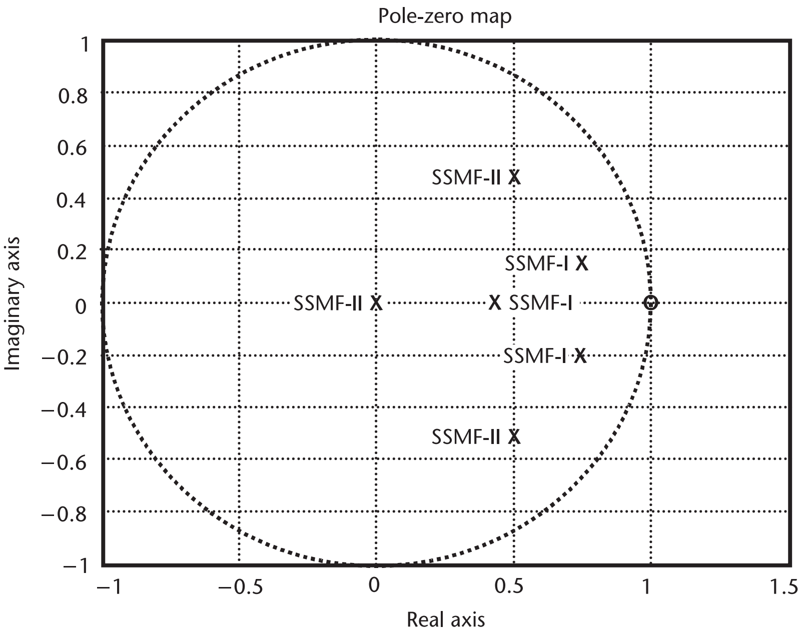


Figure 9.38 Noise transfer function pole-zero plot of SSMF-I and SSMF-II ΣΔ modulators. Each modulator has three zeros at $z = 1$, and SSMF-II has one pole at $z = 0$.

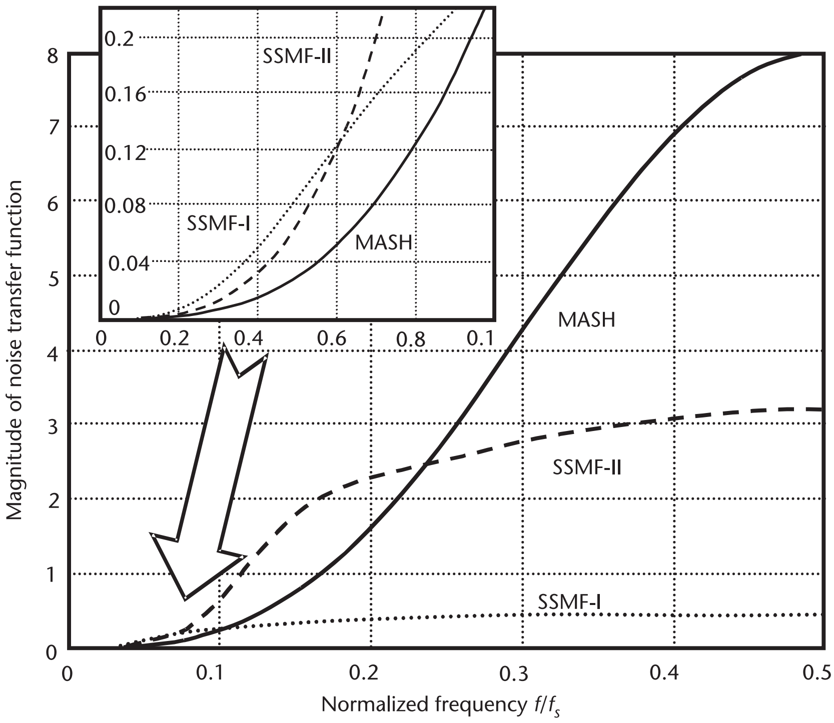


Figure 9.39 Noise transfer function magnitudes for MASH, SSMF-I, and SSMF-II ΣΔ modulators.

while the SSMF-II noise transfer function has a passband gain of 3.2, and the SSMF-I noise transfer function has a passband gain of 0.5. The highpass 3-dB corner frequency is $0.18f_s$ for the SSMF-II modulator and $0.14f_s$ for the SSMF-I modulator.

9.3.7 Phase Noise Due to ΣΔ Converters

Discrete fractional spurs become more like random noise after ΣΔ noise shaping. The SSB phase noise of the noise-shaped fractional spurs can be analyzed as follows.

According to (9.5), the 1-bit quantization error power is $\Delta^2/12$. For a quantization step size $\Delta = 1$, which is the case for a truncated binary word, the quantization error power is 1/12. This error power is spread over the sampling bandwidth or, equivalently, the reference bandwidth of $f_r = 1/T_s$. Thus, the error PSD becomes $1/12f_r$. Considering the noise shaping with an mth-order MASH ΣΔ modulator as expressed in (9.42), the frequency noise PSD is obtained as

$$S_\Omega(z) = \frac{|(1-z^{-1})^m f_r|^2}{12 f_r} = \frac{1}{12}(1-z^{-1})^{2m} f_r \qquad (9.74)$$

where the subscript Ω denotes the frequency fluctuations referred to the *input* of the divider. In order to obtain the phase fluctuations, consider the relationship between frequency ω and phase ϕ,

$$\omega(t) = \frac{d\phi(t)}{dt} \approx \frac{\phi(t) - \phi(t - T_s)}{T_s} \tag{9.75}$$

and its z domain representation,

$$2\pi \cdot \Omega(z) = \frac{\Phi(z)(1 - z^{-1})}{T_s} \tag{9.76}$$

where $T_s = 1/f_r$ is the sample period. Rearranging this expression yields

$$\Phi(z) = \frac{2\pi \cdot \Omega(z)}{f_r(1 - z^{-1})} \tag{9.77}$$

Noting that $S_\Omega(z)$ is given in terms of power, the double-sideband phase noise PSD is obtained as

$$\begin{aligned} S_\Phi(z) &= S_\Omega(z)\frac{(2\pi)^2}{|1 - z^{-1}|^2 f_r^2} = \frac{(2\pi)^2}{|1 - z^{-1}|^2 f_r^2} \cdot \frac{1}{12}(1 - z^{-1})^{2m} f_r \\ &= \frac{(2\pi)^2}{12 f_r} \cdot (1 - z^{-1})^{2m-2} \end{aligned} \tag{9.78}$$

where the subscript Φ denotes phase fluctuations. Noting that

$$(1 - z^{-1}) = |1 - e^{-j\omega T}| = 2\sin\left(\frac{\omega T}{2}\right) = 2\sin\left(\frac{\pi f}{f_r}\right) \tag{9.79}$$

and that the relationship between phase noise φ^2 in rad^2/Hz and phase noise PN in dBc/Hz is

$$\mathrm{PN}_{\mathrm{SSB}}(f)\ [\mathrm{dBc/Hz}] = 10\log\frac{\varphi^2_{\Sigma\Delta}(f)}{2}\ [\mathrm{rad}^2/\mathrm{Hz}]$$

the SSB phase noise PSD in the frequency domain is given by

$$\begin{aligned} \frac{\varphi^2_{\Sigma\Delta}(f)\ [\mathrm{rad}^2/\mathrm{Hz}]}{2} &= \frac{(2\pi)^2}{24 f_r} \cdot \left[2\sin\left(\frac{\pi f}{f_r}\right)\right]^{2(m-1)} \\ \mathrm{PN}(f)\ [\mathrm{dBc/Hz}] &= 10\log\left\{\frac{(2\pi)^2}{24 f_r} \cdot \left[2\sin\left(\frac{\pi f}{f_r}\right)\right]^{2(m-1)}\right\} \end{aligned} \tag{9.80}$$

where f is the offset frequency, and f_r is the reference sampling frequency. Now, to find the effect of this phase noise on the PLL output, an analysis similar to that performed in Section 3.7 must be performed. This noise from the ΣΔ is injected into the system as shown in Figure 9.40. Therefore, its noise transfer function to the output is given by

$$\frac{\varphi_{\text{noise\_out}}(s)}{\varphi_{\Sigma\Delta}(s)} = \frac{\dfrac{F(s)K_{\text{VCO}}K_{\text{phase}}}{N}}{s + \dfrac{F(s)K_{\text{VCO}}K_{\text{phase}}}{N}} \tag{9.81}$$

This equation is very similar to the in-band noise transfer function derived in Chapter 3.

Note that, due to the highpass nature of the ΣΔ noise transfer function, the order of the loop roll-off is very important. Recall that the noise-shaping slope of an mth-order MASH ΣΔ modulation is $20(m - 1)$ dB/dec according to (9.80), while an nth-order lowpass loop filter has a roll-off slope of $20n$ dB/dec. Therefore, the order of the loop filter must be higher than or equal to the order of the ΣΔ modulator in order to attenuate the out-of-band noise due to ΣΔ modulation. Thus, for instance, when calculating the effect of the ΣΔ modulator on out-of-band noise on the typical loop considered in Chapter 3, it is necessary to include the effect of C_2 in the loop filter, as this will provide extra attenuation out of band. In this case, the ΣΔ noise transfer function to the output will be

$$\frac{\varphi_{\text{noise\_out}}(s)}{\varphi_{\Sigma\Delta}(s)} = \frac{K_{\text{VCO}}K_{\text{phase}}(1 + sC_1R)}{s^2N(C_1 + C_2)(1 + sC_sR) + K_{\text{VCO}}K_{\text{phase}}(1 + sC_1R)} \tag{9.82}$$

where

$$C_s = \frac{C_1C_2}{C_1 + C_2}$$

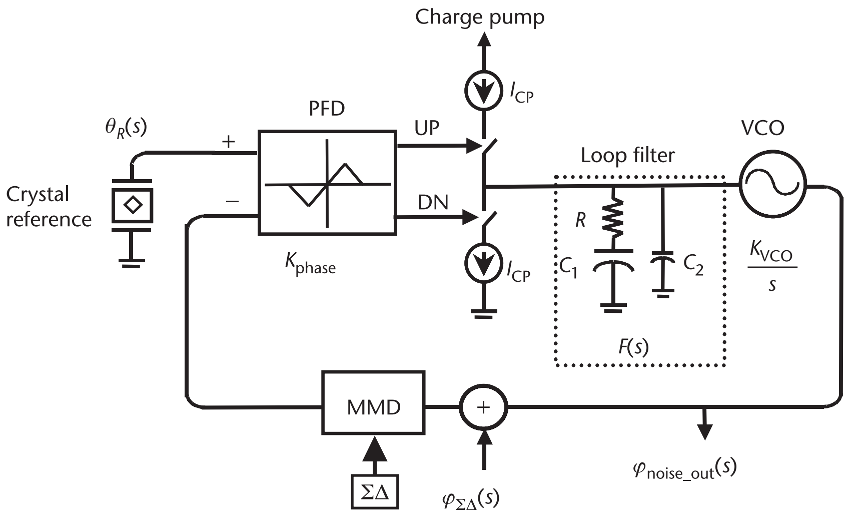


Figure 9.40 A typical fractional-*N* frequency synthesizer with a ΣΔ phase noise source added.

Example 9.3: Determining the ΣΔ Order Required for Fractional-N Synthesizer Designs

Consider the synthesizer originally designed in Examples 3.4 and 3.5. If the fractional-*N* design is to be controlled by a MASH ΣΔ modulator, find the minimum order of the MASH ΣΔ modulator such that the phase noise performance of the design will not be compromised.

Solution: First, recall that the reference frequency of the design considered in Chapter 3 was 40 MHz. Thus, the raw ΣΔ phase noise as predicted by (9.80) is given in Figure 9.41.

Note that this example considers only up to the third order at the beginning because, if a higher-order ΣΔ modulator is needed, then a more complicated, higher-order loop filter will also be required to attenuate the out-of-band phase noise effectively. Next, the raw phase noise is applied to the transfer function in (9.82). Note that all values for this formula are taken from the previous examples in Chapter 3. The phase noise from the ΣΔ is compared to the previous total phase noise predicted by this loop in order to determine the effect on overall noise performance. The results of these calculations are shown in Figure 9.42. From this plot, it is easy to see that a first-order modulator (a fractional-*N* accumulator without noise shaping) produces far too much noise and will in fact completely dominate the noise performance of this synthesizer. On the other hand, a second-order MASH ΣΔ modulator will not degrade the in-band noise. However, in the range of 1 to 3 MHz, it will increase the phase noise of the design by 3 dB as the ΣΔ noise is about equal to the total noise. It is also interesting to study the shape of the filtered ΣΔ noise curve. In-band, the ΣΔ phase noise rises at 20 dB/dec due to the second-order noise-shaping effect. At the loop corner frequency, it becomes

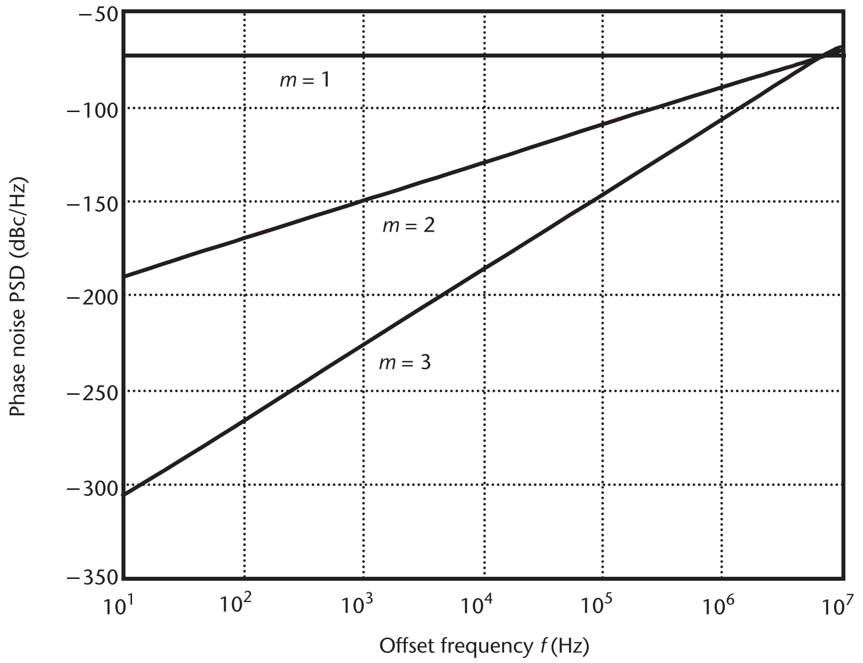


Figure 9.41 Phase noise PSD for MASH ΣΔ modulator with a 40-MHz reference frequency.

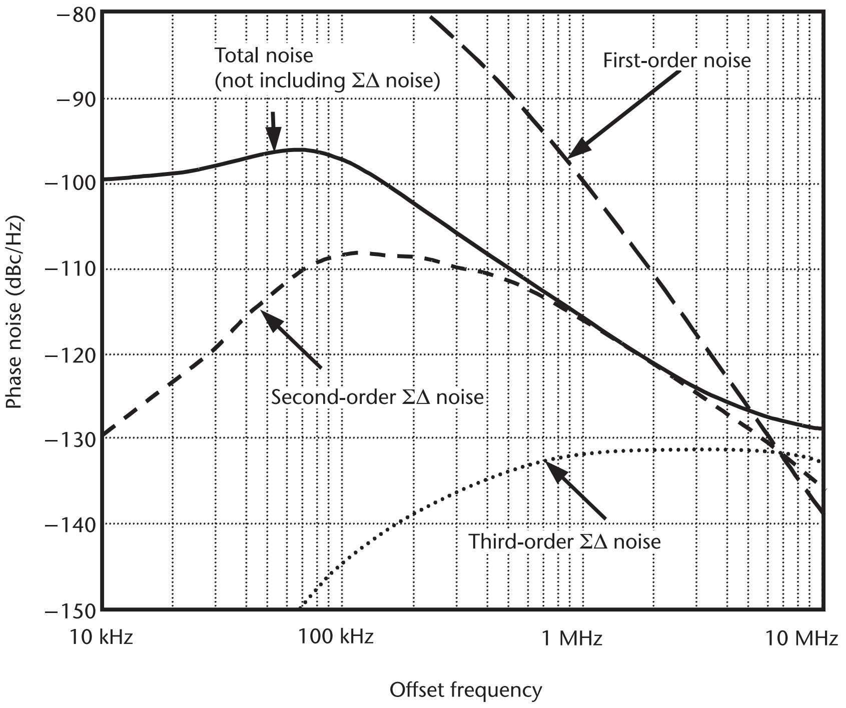


Figure 9.42 Calculated effect of ΣΔ phase noise on overall loop phase noise for various orders of ΣΔ modulators.

flat due to the attenuation of the first loop filter pole. Once the second pole from the loop filter begins to take effect, the filtered ΣΔ phase noise response falls at 20 dB/dec. The third-order ΣΔ modulator, on the other hand, has its noise well below that of the other components in the loop and, therefore, has a negligible effect on the total PLL noise. However, note that even after the second pole in the loop filter, this noise is only flat out of band. If this noise performance is not acceptable and a fourth-order ΣΔ modulator is required, then a higher-order loop filter will be needed to keep the out-of-band ΣΔ noise from growing.

It should be pointed out that the above noise PSD only models the random noise produced by the quantizer; therefore, discrete spurs can be expected to be larger than predicted by this formula. Since there is no ΣΔ noise shaping in the first-order modulator, the output of the accumulator contains discrete spurs rather than randomized noise. The phase noise curve for the first-order modulator given in Figure 9.42 will be valid if dithering is used in the fractional accumulator. As shown in Figure 9.41, the slope of the phase noise PSD is $20(m - 1)$ dB/dec. Thus, there will be no noise-shaping if only one accumulator is used. On the other hand, the phase noise PSD is shaped with slopes of 20 and 40 dB/dec for two and three loops, respectively. Also notice that for every doubling of the reference frequency, which is equivalent to doubling the sampling frequency of the ΣΔ accumulators, the in-band phase noise PSD due to fractional spurs is reduced by $6(m - 1)$ dB. Note that the noise-shaping slope for fractional *frequency* errors in (9.42) and the noise-shaping slope for fractional *phase* noise PSD in (9.80) are different. During the frequency-to-phase conversion, an integration term $(1 - z^{-1})^{-1}$ is included. In other words, the phase is obtained by integrating the frequency. Phase integration averages the frequency variation; thus, the phase noise PSD has a lower noise-shaping slope than the frequency-error-shaping curve.

9.3.8 Randomization by Noise-Shaped Dithering

The phase error PSD given in (9.80) is derived based on a uniform quantization model. For the first-order ΣΔ modulator, the accumulator output spectrum is highly dependent upon its dc input. When only input bits near the MSB are nonzero (e.g., K/F = 1/4, 1/2, 3/4), the accumulator output cycle repeats often, resulting in insufficient randomness to decorrelate the quantization error. In this case, the uniform quantization model is not quite appropriate. To randomize the tonal fractional accumulator output, a *pseudo random bit sequence* (PRBS) generator, with equal probability of ones and zeros, can be employed to dither the accumulator input value, as demonstrated in Figure 9.43. The PRBS output selectively adds an equal number of 1's and −1's to the carry-in of the adder. While it does not affect the average accumulator input value, it does decorrelate the accumulator output. Without dithering, the quantizer produces highly correlated errors that create harmonic-distortion components without additional energy in between.

For multiloop ΣΔ modulators, the inputs of all accumulators except the first accumulator are quasirandom. The input of the first accumulator is normally a constant dc value for fractional-*N* synthesis. Taking the output of the first accumulator as its input, the second loop has a nonconstant input. This is also true for any higher-order loops. Thus, the quantization error for a higher-order multiloop ΣΔ modulator is approximately random and decorrelated and fits the uniform quantization model well. Thus, for higher-order ΣΔ modulators, dithering is less important, although dithering is still useful to remove tones.

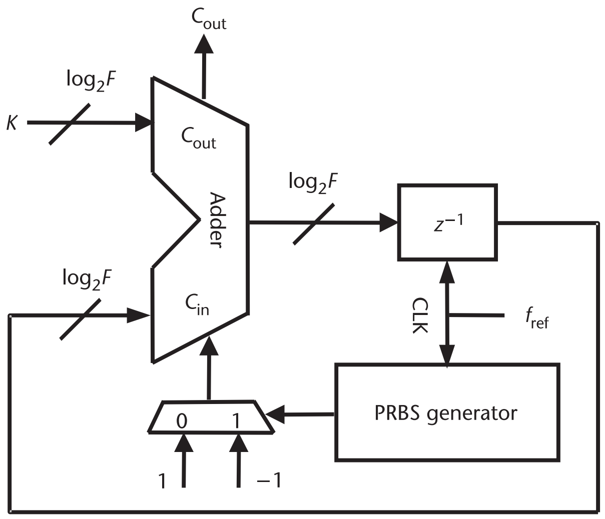


Figure 9.43 A fractional accumulator with dithering scheme.

In the above dithering configuration, a dither sequence is added at the input of the modulator, which decorrelates the ΣΔ modulator output and spreads the fractional tone energy over the entire band without filtering. Input dithering without noise shaping can potentially degrade the in-band rms noise. A better solution is to noise-shape the dithering output spectrum such that the dither energy is pushed out of the signal band. The highpass filtered dithering can be implemented by adding the dither sequence to the quantizer input, *not* to the modulator input. For a single-bit ΣΔ modulator, the dither magnitude is as large as the maximum input, which may seriously overload the quantizer. On the other hand, dithering the ΣΔ modulator input in the signal band can normally be treated as a perturbation to the input frequency word. For a multistage ΣΔ modulator, each modulator can have a dedicated dithering circuit to provide the optimal decorrelation of the quantization noise. Since the quantization errors in higher-order loops are more random, dithering the first modulator is more important than dithering the higher-order modulators. In order to make the dither transfer function of the entire ΣΔ modulator have the same shape as that of the noise transfer function, dither sequences need to be prefiltered in all modulators except the last one. Figure 9.44 illustrates a third-order MASH ΣΔ modulator with noise-shaped dithering schemes. Thus, in this case, the output is given by

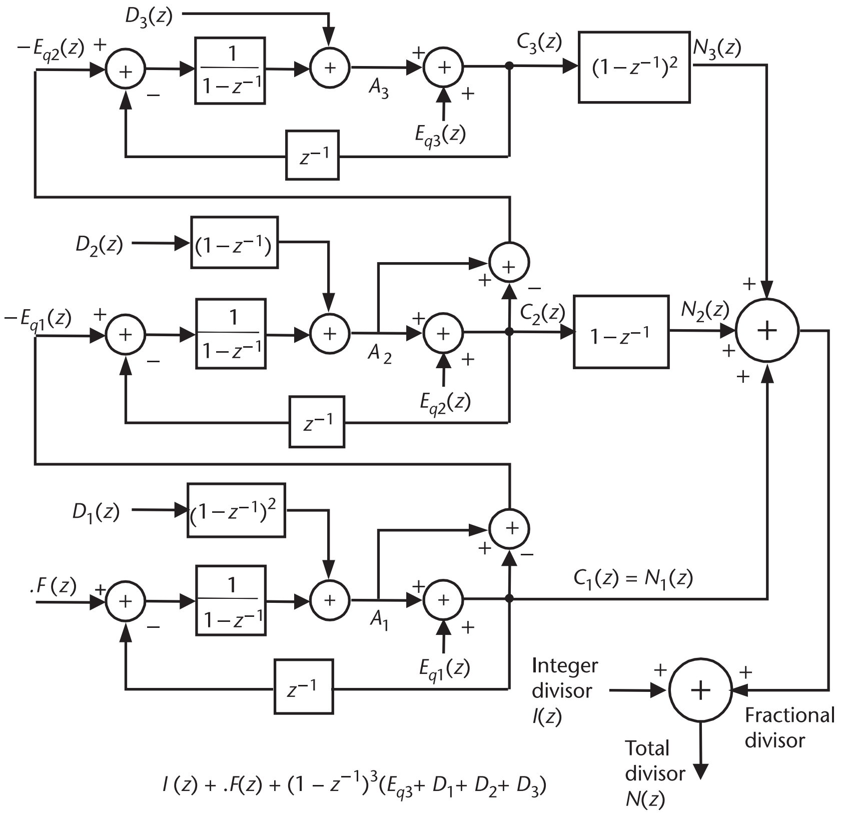


Figure 9.44 A third-order MASH ΣΔ modulator with noise-shaped dithering schemes.

$$\begin{aligned}
N_1(z) &= .F(z) + (1 - z^{-1})E_{q1}(z) + (1 - z^{-1})^3 D_1(z) \\
N_2(z) &= (1 - z^{-1})[-E_{q1}(z) + (1 - z^{-1})E_{q2}(z) + (1 - z^{-1})^2 D_2(z)] \\
N_3(z) &= (1 - z^{-1})^2[-E_{q2}(z) + (1 - z^{-1})E_{q3}(z) + (1 - z^{-1})D_3(z)] \\
N(z) &= I(z) + N_1(z) + N_2(z) + N_3(z) \\
&= I(z) + .F(z) + (1 - z^{-1})^3[E_{q3}(z) + D_1(z) + D_2(z) + D_3(z)]
\end{aligned} \tag{9.83}$$

As shown, the above dithering process not only decorrelates the quantization energy but also moves the decorrelated noise energy to higher frequencies, where it can easily be filtered.

9.3.9 Spur Reduction Using Precalculated Seeds

To avoid artificial spurs at the synthesizer output, ΣΔ accumulators can be loaded with precalculated start values. It is known that any repetition of time sequence will cause artificial spurs in the spectrum with a frequency inversely proportional to the repetition's period. Loading different seed values to different accumulators can break the repetition in the time domain. Thus, artificial spurs can be reduced using a proper start value for each accumulator. In addition, a different order of ΣΔ noise shaper can be selected simultaneously. Referring to the MASH 1-1-1 ΣΔ modulator shown in Figures 9.21 and 4.13, resetting the first accumulator with a loaded seed is more important than for the second and the third accumulators, since only the first accumulator has a constant input, which may result in a repeated accumulator value (i.e., artificial spurs). The second and third accumulators take the outputs from previous accumulators as their inputs, which are more like random numbers. Thus, the values in the second and the third accumulators are not likely to be repeated. The proposed spur-reduction scheme with precalculated accumulator seed values can be simply implemented using the existing accumulator reset, and, hence, does not require additional hardware. In contrast, other spur-randomization schemes, such as dithering the LSB of the ΣΔ input, not only require additional hardware but also cause large frequency variation for small input words.

Spur reduction with precalculated seeds in ΣΔ accumulators has been demonstrated in measurement in [5], which presents a multiband WLAN fractional-*N* frequency synthesizer with a programmable MASH ΣΔ modulator. Figure 9.45 shows the measured worst spur levels for WLAN 802.11b channels, which correspond to the worst fractional spurs in the design with a fractionality of 1/30. The measured spurs clearly demonstrate the effect of spur reduction using precalculated seeds.

9.3.10 Dynamic Range

A ΣΔ modulator design involves optimization and trade-offs of multiple architectural parameters, such as the number of output bits, the number of feedback bits, the order of the modulator, the oversampling ratio, and the input range. The output

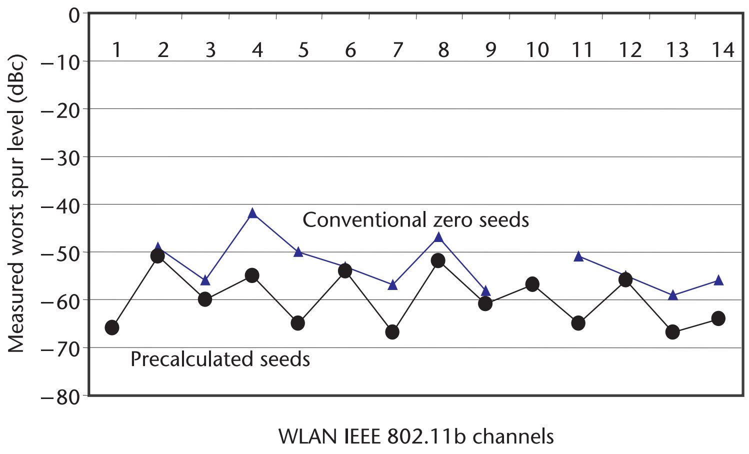


Figure 9.45 Measured worst spur levels for 802.11b channels with precalculated seeds (bottom curve) and conventional zero seeds (top curve).

SNR of a ΣΔ modulator is dependent upon not only the number of output bits but also the input level. When the input level is low, the modulator output SNR is linearly proportional to the input level, as shown in Figure 9.46. However, as with any nonlinear system, the input level cannot be increased infinitely. The *dynamic range* of a ΣΔ modulator is defined as the maximum SNR achievable for a certain topology [11]. The *overload level* of a ΣΔ modulator is defined as the maximum input magnitude for which the ΣΔ modulator still operates properly. A ΣΔ modulator will continue to operate properly provided the SNR is not degraded more than 6 dB from the dynamic range value, as illustrated in Figure 9.46. When the input level increases beyond the overload level, the modulator output SNR

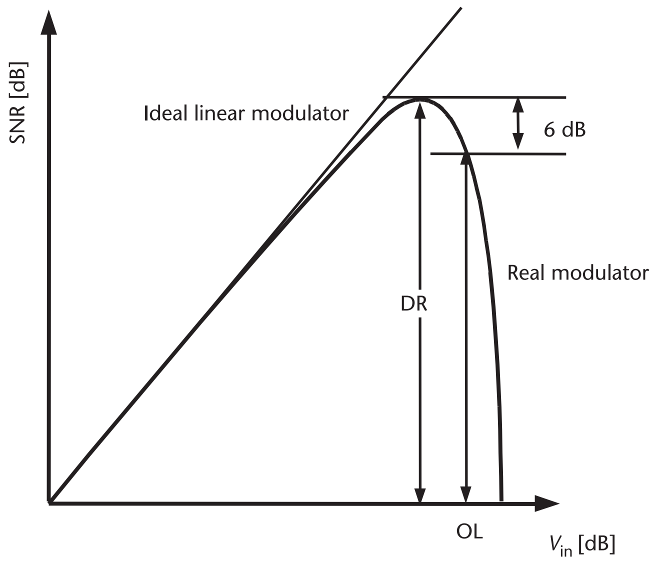


Figure 9.46 Dynamic range and overload of a ΣΔ modulator.

decreases quickly. As discussed before, the quantization noise can be considered as random white noise with a uniform amplitude distribution in the time domain between $-\Delta/2$ and $\Delta/2$, and the quantization noise power is given by $\Delta^2/12$. According to (9.56), the quantization noise due to a fractional-*N* mechanism is noise shaped by the noise transfer function of the ΣΔ modulator. Assuming the noise transfer function is of the form of a MASH ΣΔ modulator, that is, $H_e(z) = (1 - z^{-1})^n$, the shaped in-band quantization noise power can be derived from (9.33) and (9.5) as

$$n_0^2 = \frac{2}{f_s}\int_0^{f_0} \frac{\Delta^2}{12}|1 - z^{-1}|^{2n}\, df \approx \frac{\Delta^2 \pi^{2n}}{12(2n+1)}\left(\frac{1}{\text{OSR}}\right)^{2n+1} \tag{9.84}$$

where $z = \exp(j2\pi f/f_s)$. Considering a sinusoidal signal with an amplitude of A_x normalized to $\Delta/2$, the signal power is given by

$$S = \frac{1}{2}A_x^2\left(\frac{\Delta}{2}\right)^2 \tag{9.85}$$

Hence, the SNR at the quantizer output can be obtained as

$$\text{SNR} = \frac{S}{n_0^2} = \frac{3\pi}{2}A_x^2(2n+1)\left(\frac{\text{OSR}}{\pi}\right)^{2n+1} \tag{9.86}$$

Assuming the quantizer has B output bits, the maximum full-scale amplitude it can represent is

$$A_x = 2^B - 1 \tag{9.87}$$

Therefore, the dynamic range of the ΣΔ modulator, which is the maximum achievable SNR at the modulator output, can be obtained as

$$\text{DR} = \text{SNR}_{\max} = \frac{3\pi}{2}(2^B - 1)^2(2n+1)\left(\frac{\text{OSR}}{\pi}\right)^{2n+1} \tag{9.88}$$

It is evident that the dynamic range of a ΣΔ modulator is dependent upon the number of quantizer output bits, the order of the modulator, and the oversampling ratio. The multibit topology directly improves the dynamic range by $20\log(2^B - 1)$ dB compared to that achieved using a single-bit topology. Therefore, the multibit ΣΔ modulator is critical for applications with a low oversampling ratio, while a single-bit ΣΔ modulator is often used with high OSR, where the quantization noise is fairly low due to a high oversampling ratio.

9.3.11 Maximal Loop Bandwidth

The loop bandwidth f_c is limited when a ΣΔ modulator is employed. Increasing the loop bandwidth will degrade the phase noise due to ΣΔ-shaped fractional spurious components, although it is often desirable to reduce other noise sources, such as the VCO phase noise, by widening the loop bandwidth. If the synthesizer phase noise cannot exceed some specified amount $\varphi_{\max}(f)$, the maximum loop bandwidth f_c can be determined. First, note that (9.80) can be approximated as

$$\frac{\varphi^2_{\Sigma\Delta}(f)}{2} = \frac{(2\pi)^2}{24 f_r} \cdot \left(\frac{\pi f}{f_r}\right)^{2(m-1)} \tag{9.89}$$

by noting that $\sin x \approx x$. Next, assume for simplicity, that the loop noise transfer function given in (9.81) can be approximated as

$$\frac{\varphi_{\text{noise\_out}}(s)}{\varphi_{\Sigma\Delta}(s)} = \left(\frac{f_c}{f}\right)^n \tag{9.90}$$

where f_c is the loop bandwidth for an nth-order loop. This approximation is only valid beyond the corner frequency of the loop f_c. Thus, the output phase noise from the ΣΔ modulator can be approximated as

$$\frac{\varphi^2_{\text{noise\_out}}(f)}{2} = \left(\frac{f_c}{f}\right)^{2n} \frac{(2\pi)^2}{24 f_r} \cdot \left(\frac{\pi f}{f_r}\right)^{2(m-1)} \tag{9.91}$$

If this cannot exceed $\varphi_{\max}(f)/2$, then f_c must be less than

$$f_c < \left[\frac{\varphi^2_{\max}(f)}{2} \frac{24 f_r}{(2\pi)^2} \cdot \left(\frac{f_r}{\pi f}\right)^{2(m-1)}\right]^{(1/2n)} \cdot f \tag{9.92}$$

If the required PLL phase noise is assumed to be flat with value A_n [rads^2/Hz] inside the loop bandwidth and zero outside the loop bandwidth, as shown in Figure 9.47, and the loop transfer function inside the loop bandwidth can be assumed to be unity, then the maximum loop bandwidth can be estimated.

The integrated rms phase noise in [rms rad] or [rms deg] is often quoted for synthesizer phase noise evaluation. The SSB phase noise PSD (A_n) is related to the integrated rms phase noise by

$$\text{IntPN}_{\text{rms}} = \sqrt{\int_{-f_c}^{f_c} A_n \, df} = \sqrt{2 f_c A_n} \tag{9.93}$$

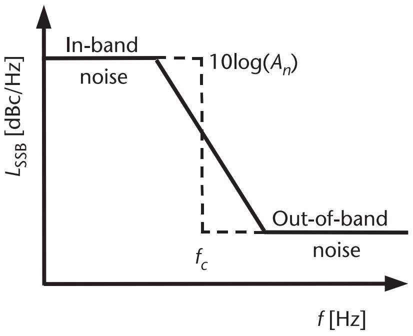


Figure 9.47 Simplified synthesizer output phase noise PSD.

as discussed in Chapter 3. Similarly, the integrated rms frequency noise Δf_n can be found. Recall that frequency is the derivative of phase, and the frequency noise PSD can be found from the phase noise PSD as $A_n \cdot f^2$. Note that $\sqrt{A_n} \cdot f$ converts the phase noise amplitude to frequency noise amplitude, where multiplying by f is equivalent to taking a derivative in the frequency domain. Therefore, the integrated rms frequency noise Δf_n in [rms Hz] within noise bandwidth f_c can be approximated by [10]:

$$\Delta f_n \approx \sqrt{\int_{-f_c}^{f_c} A_n f^2 \, df} = \sqrt{\frac{2}{3} A_n f_c^3} \tag{9.94}$$

The fractional-*N* scheme with ΣΔ noise shaping dithers the loop division ratio, varying the frequency or phase of the MMD output constantly. In the frequency domain, the dynamic range of a synthesizer is the ratio of the largest possible frequency variation to the integrated rms frequency noise Δf_n [12]. The largest frequency variation is given by $\Delta N \cdot f_r$, where $\Delta N = 2^B - 1$ is the largest modulus variation, and B is the number of modulator output bits. Hence, the dynamic range of the PLL corresponding to the phase noise PSD illustrated in Figure 9.47 is given by

$$\mathrm{DR}_{\mathrm{PLL}} = \left(\frac{\Delta N \cdot f_r}{\Delta f_n}\right)^2 \tag{9.95}$$

To prevent the synthesizer spectrum purity from being corrupted by the phase noise due to fractional spurs, the dynamic range of the ΣΔ modulator has to be larger than the synthesizer PLL dynamic range, given by

$$\mathrm{DR}_{\Sigma\Delta} = \frac{3\pi}{2}(2^B - 1)^2 (2n + 1)\left(\frac{\mathrm{OSR}}{\pi}\right)^{2n+1} > \left(\frac{\Delta N \cdot f_r}{\Delta f_n}\right)^2 \tag{9.96}$$

where the ΣΔ modulator dynamic range expressed in (9.88) is used.

Considering (9.94) and for $\text{OSR} = f_r/2f_c$, the in-band phase-noise-limited maximal loop bandwidth associated with an *n*th-order MASH ΣΔ modulator with *B* output bits is obtained as

$$f_c < \frac{f_r}{2\pi} \cdot \left[\frac{f_r}{8\pi^2} \cdot A_n(2n+1)\right]^{\frac{1}{2n-2}} \tag{9.97}$$

It should be pointed out that the above result provides only an approximate upper bound of the loop bandwidth limited by the integrated phase noise and the use of a MASH ΣΔ modulator. In practice, the required loop bandwidth is narrower than the above-predicted value since the quantization noise of the ΣΔ modulator is reduced after the high-frequency pole of the PLL. If Butterworth poles are added to damp the high-frequency noise, the in-band quantization noise power is also increased. For a single-stage, third-order, ΣΔ modulator (SSMF-II), as shown in the MASH ΣΔ modulator of Figure 9.36, the in-band phase noise power is about four times higher than that of a third-order MASH ΣΔ modulator. Therefore, maximal loop bandwidth for a single-loop, multibit ΣΔ modulator should be smaller than 25% of the value calculated based on (9.97). For instance, if the in-band phase noise A_n is –100 dBc/Hz, the above equation estimates the maximal loop bandwidth as 1 MHz for a third-order ΣΔ modulator with $f_r = 40$ MHz. In practice maximal loop bandwidth of $f_c < 250$ kHz should be chosen.

9.3.12 Optimal Parameters

As shown in previous sections, different ΣΔ modulator topologies offer not only different noise transfer functions but also have different dynamic range and stability constraints. The design goal is to find an optimal set of parameters that can lead to the maximal dynamic range without causing stability and overload problems. For a single-stage ΣΔ modulator topology as shown in Figure 9.48, the noise transfer function at low frequency can be approximated by [11]:

$$|H_e(z)| = \frac{1}{1 + \sum_{i=1}^{n} \prod_{j=i}^{n} a_j k \left(\frac{z^{-1}}{1 - z^{-1}}\right)^{n-i+1}} \approx \frac{\left|1 - z^{-1}\right|^n}{\prod_{i=1}^{n} a_i k} \tag{9.98}$$

where a_i is the coefficient of the *i*th integrator, and *k* is the quantizer gain. $a_n k$ is thus the effective coefficient of the last integrator. The equation shows that increasing the loop parameters reduces the in-band noise power, which benefits the output SNR and dynamic range. However, increasing the loop parameters worsens the loop

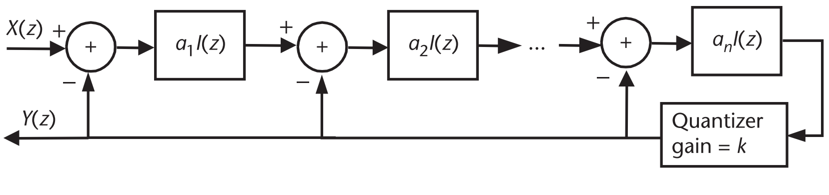


Figure 9.48 Optimal coefficients for single-stage ΣΔ modulator.

stability and the overload level. Therefore, it is desirable to find the optimal combination of loop coefficients that correspond to the maximal dynamic range without going into the overload range. However, it is a rather difficult task to find the optimal loop parameters analytically. Extensive SNR and dynamic range simulations have been performed for second- to fourth-order, single-loop ΣΔ modulators in [11]. It is concluded that the dynamic range of a single-loop ΣΔ modulator with a single-bit quantizer is maximized when the product of all effective integrator coefficients is approximately given by

$$\prod_{i=1}^{n} a_i k = 1, \frac{1}{5}, \frac{1}{25} \quad n = 2, 3, 4 \tag{9.99}$$

where the nth term $a_n k = 2$. To reduce the internal error accumulation, the loop coefficients should increase from the first to the last integrator. Moreover, for stability reasons, $a_i < 3/4$, where $i = 1, 2, \ldots, n - 1$, should be chosen. For multibit topologies, due to intrinsically better stability, the integrator coefficients can be increased, which leads to lowered in-band noise and increased SNR and dynamic range. If a 4-bit quantizer is used in the single-stage ΣΔ modulator shown in Figure 9.48, the dynamic range is maximized when the product of all effective integrator coefficients is approximately given by

$$\prod_{i=1}^{n} a_i = \frac{27}{16}, \frac{27}{32}, \frac{27}{128} \quad n = 2, 3, 4 \tag{9.100}$$

If a different number of quantizer bits or a different topology is used, the above optimized parameters need to be redetermined by simulation. For instance, simulation has demonstrated that a multiloop ΣΔ modulator of order higher than two always offers better dynamic range than a single-loop ΣΔ modulator of the same order.

9.3.13 Performance Comparison

Finally, Table 9.1 summarizes the performance comparison for a few third-order ΣΔ modulators discussed in this chapter. As shown, the MASH 1-1-1 ΣΔ modulator provides the simplest implementation with the fastest speed. With proper buffering (as illustrated in Figure 4.13), the maximum delay of a MASH 1-1-1 modulator can be just one K-bit adder delay, where K is the number of input bits to the modulator. However, a MASH 1-1-1 ΣΔ modulator does not provide high-frequency attenuation and is very tonal at low frequency [13]. The single-stage with a single feedback modulator shown in Figure 9.26 is capable of outputting any number of bits greater than three. The SSMF-I modulator shown in Figure 9.35 provides the best high-frequency attenuation with the trade-off of the worst in-band noise shaping. The SSMF-II modulator shown in Figure 9.37 provides sufficient high-frequency attenuation with acceptable in-band noise shaping and is faster than the structure shown in Figure 9.36 due to the buffers in the accumulator signal paths. Choosing a proper ΣΔ modulator topology is important and dependent upon system requirements.

Table 9.1 Comparison of the Third-Order ΣΔ Modulators

ΣΔ Topology	*Close-In Noise Shaping*	*High-Frequency Attenuation*	*Modulator Artificial Tones*	*Output Bits*	*Speed*	*Hardware Area*
MASH 1-1-1 (Figure 9.21)	Best	Worst	Many	4 bits (–3~4)	Fastest	Smallest
MASH 1-2 (Figure 9.22)	Same as MASH 1-1-1	Same as MASH 1-1-1	Some	3 bits (–1~2)	Fast	Small
Single-stage with single feedback (Figure 9.26)	Same as MASH 1-1-1	Same as MASH 1-1-1	Some	Any number of bits greater than 3	Fast	Small
SSMF-I (Figure 9.35)	Worst	Best	Few	Any number of bits	Slow	Large
SSMF-II (Figure 9.36)	Good	Good	Few	Any number of bits	Slow	Large
SSMF-II (Figure 9.37)	Good	Good	Few	Any number of bits	Fast	Large

References

[1] Norsworthy, S. R., R. Schreier, and G. C. Temes, (eds.), *Oversampling Delta-Sigma Data Converters*, New York: IEEE Press, 1992.

[2] Boser, B. E., and B. A. Wooley, "The Design of Sigma-Delta Modulation Analog-to-Digital Converters," *IEEE J. Solid-State Circuits*, Vol. 23, No. 6, December 1988, pp. 1298–1308.

[3] Wells, J. N., "Frequency Synthesizers," United States Patent No. 4609881, September 1986.

[4] Miller, B., and R. Conley, "A Multiple Modulator Fractional Divider," *Proc. 44th Annual Frequency Control Symposium*, Baltimore, MD, May 1990, pp. 559–568.

[5] Rogers, J. W. M., et al., "A Fully Integrated Multi-Band ΣΔ Fractional-*N* Frequency Synthesizer for a MIMO WLAN Transceiver RFIC," *IEEE Journal on Solid State Circuits*, Vol. 40, No. 3, March 2005, pp. 678–689.

[6] Riley, T. A., M. Copeland, and T. Kwasniewski, "Delta-Sigma Modulation in Fractional-*N* Frequency Synthesis," *IEEE J. Solid-State Circuits*, Vol. 28, May 1993, pp. 553–559.

[7] Jackson, T., G. Eapen, and F. Dai, "Feed Forward Sigma Delta Interpolator for Use in a Fractional-*N* Synthesizer," U.S. Patent Application Publication, No. 2002/0067773 A1, June 6, 2002.

[8] Chao, K., et al., "A Higher-Order Topology for Interpolative Modulation for Oversampling A/D Converters," *IEEE Trans. on Circuits and Systems*, Vol. 37, March 1990, pp. 309–318.

[9] Muer, B., and M. Steyaert, "A CMOS Monolithic ΣΔ-Controlled Fractional-*N* Frequency Synthesizer for DCS-1800," *IEEE J. Solid-State Circuits*, Vol. 37, July 2002, pp. 835–844.

[10] Rhee, W., B. Song, and A. Ali, "A 1.1-GHz CMOS Fractional-*N* Frequency Synthesizer with a 3-b Third-Order ΔΣ Modulator," *IEEE J. Solid-State Circuits*, Vol. 35, October 2000, pp. 1453–1460.

[11] Marques, A., et al., "Optimal Parameters for ΔΣ Modulator Topologies," *IEEE Trans. on Circuits and Systems-II: Analog and Digital Signal Processing*, Vol. 45, No. 9, September 1998, pp. 1232–1241.

[12] De Muer, B., and M. S. J., Steyaert, "On the Analysis of ΔΣ Fractional-*N* Frequency Synthesizers for High-Spectral Purity" *IEEE Trans. on Circuits and Systems-II: Analog and Digital Signal Processing*, Vol. 50, No. 11, November 2003, pp. 784–793.

[13] Shu, B. K., et al., "A Comparative Study of Digital Modulators for Fractional-*N* Synthesis," *8th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, Malta, Vol. 3, September 2001, pp. 1391–1394.

CHAPTER 10

Direct Digital Synthesis

10.1 Introduction

Modern communication systems are placing increasing demands on the frequency resolution, channel switching speed, and bandwidth requirements of frequency synthesis. For instance, spread-spectrum applications require a frequency synthesizer that is capable of tuning to different output frequencies with extremely fine frequency resolution and switching speed of the order of nanoseconds. The resolution and switching-speed requirements of many systems are surpassing the performance capabilities of a conventional analog phase-locked loop (PLL). The conventional PLL-based frequency synthesizer has difficulty meeting these requirements due to internal loop delay, low resolution, and the limited tuning range of the VCO. In contrast, a direct digital synthesizer (DDS) generates a digitized waveform of a given frequency by accumulating phase changes at a higher clock frequency. DDS is a digital technique for frequency synthesis, waveform generation, sensor excitation, and digital modulation and demodulation. Since there is no feedback in a DDS structure, it is capable of extremely fast frequency switching or hopping at the speed of the clock frequency. DDS provides many other advantages, including fine frequency-tuning resolution and continuous-phase switching. In addition, as will be discussed in Chapter 11, DDS can provide quadrature signals with accurate I/Q matching. DDS can directly provide various modulations. DDS can also generate arbitrary waveforms in the digital domain. The increasing availability of ultra-high-speed digital-to-analog converters (DACs) allows a DDS to operate at clock frequencies of more than 10 GHz.

The DDS has many advantages; however, it has two major deficiencies that are related to the inadequacy of the semiconductor technology. The first deficiency is that the output spectrum of the DDS is normally not as clean as the PLL output. The noise floor of the DDS output spectrum is limited by a finite number of bits in the DAC. A 12-bit DAC provides a theoretical noise floor of –72 dBc, which is much less than that of a PLL synthesizer. Normally, a –100 dBc/Hz noise floor can be achieved by a PLL synthesizer at a 100-kHz offset. The DDS also suffers from a high level of spurious output derived from the discrete phase-accumulation and phase-truncation processes, as well as the DAC nonlinearity. The second deficiency is that the DDS output frequency is limited by the maximum operation frequency of the DAC and the digital logic. Although DACs with gigahertz sampling frequencies have been reported, they normally consume a large amount of power with poor resolution. As the sampling frequency increases, the power required both for the DAC and for digital waveform computing circuitry increases approximately in proportion. Nevertheless, the DDS can be combined into a multiloop hybrid PLL arrangement where, with careful design, the best of both worlds can be achieved. The DDS gives the hybrid synthesizer small step size and faster switching, while the PLL synthesizer keeps the power consumption low, even when the desired output frequency is in the gigahertz range.

10.2 DDS Theory of Operation

A basic DDS system, as shown in Figure 10.1, consists of a numerically controlled oscillator (NCO) to generate the sampled signal, followed by a DAC used to convert the digital waveform to an analog signal. Since the DAC output is sampled at the reference clock frequency, a deglitch lowpass filter is typically used to smooth the waveform. The NCO uses an N-bit accumulator to generate a phase ramp based on the N-bit input frequency control word (FCW). A read-only memory (ROM) stores the amplitude information of the desired waveform. With the phase word as the address, the ROM's output is the amplitude word of the synthesized waveform. The FCW is continuously added to (accumulated with) the last sampled phase value by an N-bit adder. When the accumulator reaches the N-bit maximum

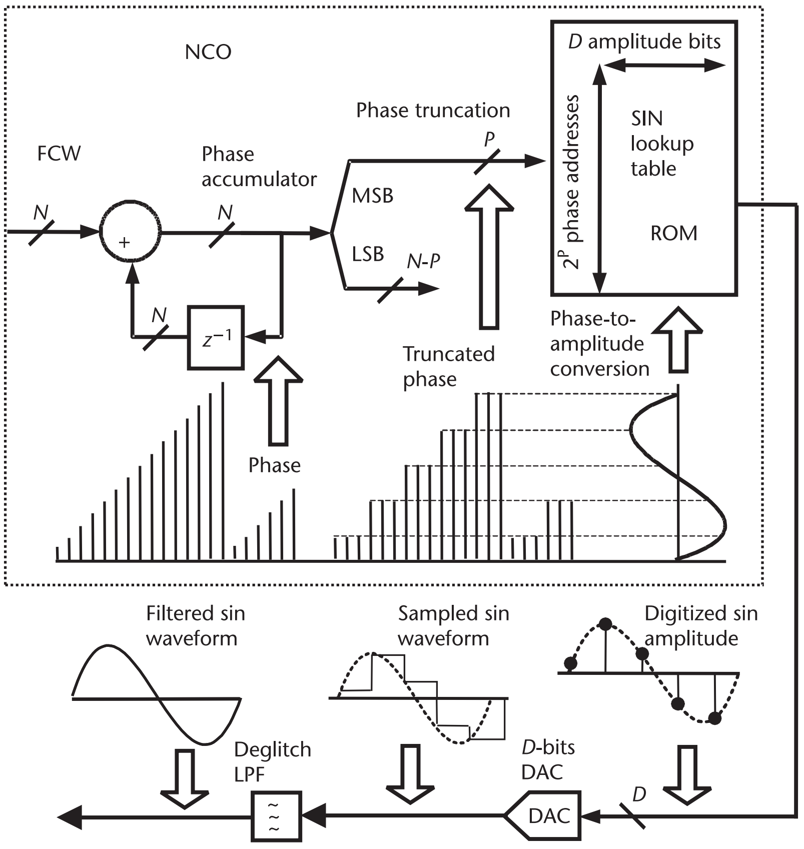


Figure 10.1 A ROM-lookup-table-based DDS Architecture.

value, the accumulator rolls over (overflows) and continues. The rollover rate of the accumulator is hence the DDS output frequency:

$$f_o = f_{\text{clk}} \frac{\text{FCW}}{2^N} \tag{10.1}$$

where f_{clk} is the DDS sample clock frequency. Since the FCW can be stepped by unity, the resolution of the DDS is given by

$$\text{Resolution} = \frac{f_{\text{clk}}}{2^N} \tag{10.2}$$

It is now evident that a DDS can achieve a very fine resolution if the accumulator size N is large. For example, if a 32-bit accumulator is used, and the DDS operates at a clock frequency of 100 MHz, its resolution is 0.0233 Hz. However, fine resolution relies on a large number of accumulator bits, which corresponds to a long phase word. Note that the ROM size is proportional to the addressing range 2^N. As a result, a large ROM lookup table is required. In order to reduce the ROM size while keeping a fine step size, only the most significant P bits of the phase word are used to address the ROM. This truncation at the accumulator output causes a quantization error that will be discussed in Section 10.3.4. The ROM size is equal to

$$\text{ROM size} = 2^P \cdot D \tag{10.3}$$

where D, the number of amplitude bits, is determined by the number of DAC input bits.

While increasing the number of phase bits is always feasible, increasing the number of DAC input bits is limited by the semiconductor technology. Even if the desired number of DAC bits can be implemented using an available technology, adding bits is costly due to large increases in die size and power consumption. Therefore, the goal of DDS design is to minimize the phase-truncation error such that the DDS output noise is dominated by the DAC quantization noise. A good design practice is that the number of truncated phase bits P should be slightly larger than the number of output amplitude bits D, which corresponds to the number of DAC input bits.

If a sinusoidal waveform is to be generated by the DDS, the frequency domain representation of the sinusoid is an impulse function at the specified frequency. The NCO, however, puts out discrete sampled data of this sinusoid at the NCO sampling clock rate. In the time domain, the NCO output is the product of the sampling clock edge strobes (impulses at the clock edges, hereafter simply called the sampling clock) multiplied by the sinusoidal waveform, which produces a stream of impulses with the sinusoid's magnitude. In the frequency domain, the sampling strobes of the reference clock produce a stream of impulses at multiples of the NCO clock frequency. Since the sampling clock is multiplied by the sinusoid in the time domain, the frequency domain components of the sinusoid and the sampling clock are convolved to produce the spectral representation of the NCO output as shown in Figure 10.2.

In a DDS system, the DAC translates the NCO digital output into an analog signal. The DAC is a sample-and-hold system that takes the NCO digital amplitude words and converts them into an analog voltage or current. For zero-order hold, the DAC holds the output value for one sample period. Figure 10.3 illustrates the temporal and spectral representations of the DDS output after the DAC and the deglitch lowpass filter. The sample-and-hold transfer function can be expressed as $u(t) - u(t - T_s)$ in the time domain, where $u(t)$ is the unit step function. In the frequency domain, the sample-and-hold function is a *sinc* function $[\sin(x)/x]$ with the first null at the sampling clock frequency f_s, as shown in Figure 10.3(a). In the time domain, the DAC output is the convolution of the sample-and-hold function and the NCO output shown in Figure 10.3(b). Since the time domain representation is convolved, the spectral representation of the DAC output is the multiplication of the sample-and-hold function and the NCO output. As a result, the DDS output has a *sinc* envelope. The DAC's zero-order sample-and-hold imposes a *sinc* attenuation envelope to the fundamental, images, and harmonics in the DDS output spectrum as

$$\text{Attenuation} = 20 \log \left[\frac{\sin\left(\frac{\pi f}{f_s}\right)}{\frac{\pi f}{f_s}} \right] \text{[dB]} \tag{10.4}$$

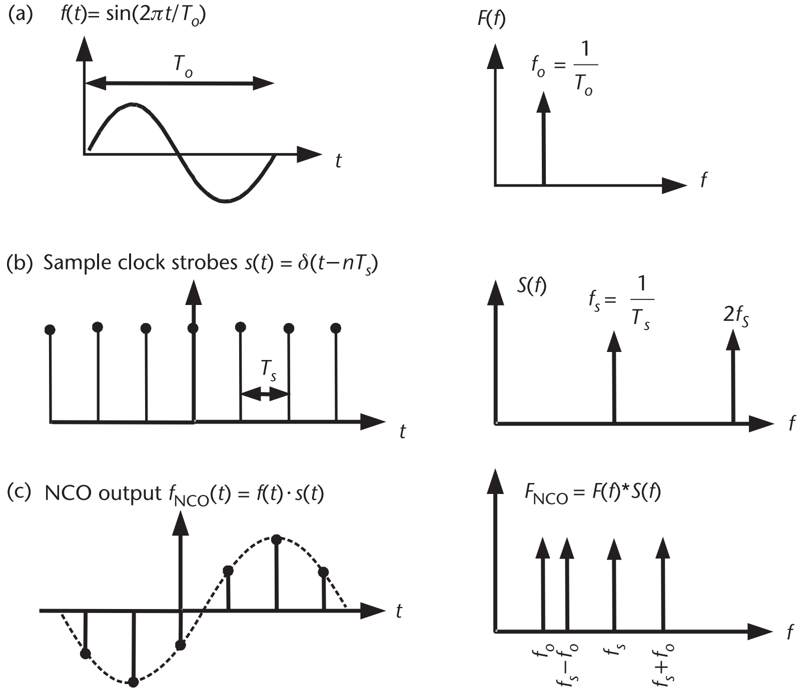


Figure 10.2 Temporal and spectral representations of (a) the sinusoidal waveform, (b) the sample clock strobes, and (c) the NCO output.

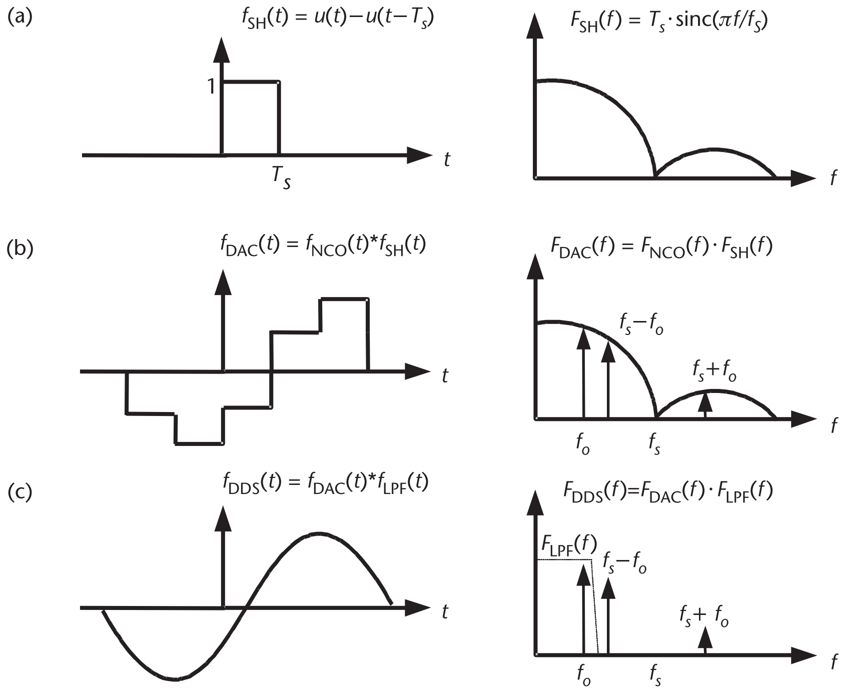


Figure 10.3 Temporal and spectral representations of (a) the sample-and-hold function, (b) the DAC output, and (c) the DDS output.

The deglitch lowpass filter removes the spurs located outside the filter passband. The DDS output is thus a pure sinusoidal wave with low spurious components, as shown in Figure 10.3(c).

10.3 DDS Spectral Purity

The DDS spectral purity is dependent upon a number of factors, including the clock phase noise, the number of phase bits applied to the sine-lookup function, the number of bits in the lookup table, the DAC errors, including nonlinearities and quantization noise, and the deglitch filter noise, as shown in Figure 10.4. If there are no phase and amplitude truncations and if the DAC is an ideal linear device with an infinite number of input bits, the DDS output will be an ideal sinusoidal waveform expressed as

$$s(t) = A \sin\left(\frac{2\pi \cdot \mathrm{FCW}}{2^N} \cdot \frac{t}{T_{\mathrm{clk}}}\right) = A \sin\left(\frac{2\pi \cdot \mathrm{FCW}}{2^N} \cdot n\right) \tag{10.5}$$

where A is the full-scale output magnitude of the DAC, and $t = nT_{\mathrm{clk}}$, where T_{clk} is equal to T_s, the sampling period. However, a DDS is not an ideal device. It has four principal noise and spurious sources: the reference clock (E_{CLK}), phase truncation (E_P), amplitude truncation (E_A), and DAC nonlinearities (E_{DAC}). For an NCO without the DAC, the quantization noise and spurs are mainly caused by

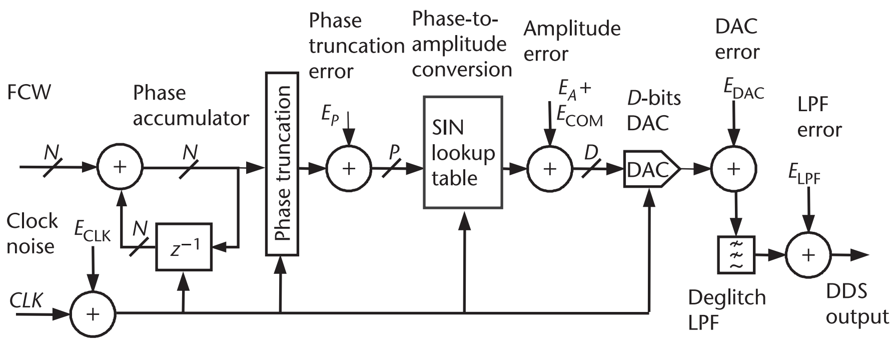


Figure 10.4 DDS noise and spurious sources.

two nonlinear operations: (1) the truncation of the phase accumulator output bits in order to reduce the ROM size, and (2) the finite precision of the sinusoidal magnitudes stored in the ROM using a finite number of bits. In order to reduce the ROM size, various ROM compression algorithms have been developed. ROM compression also causes amplitude error, E_{COM}, that adds to E_A the error produced by amplitude truncation, as shown in Figure 10.4. For DDS, the nonlinearity and additional noise due to the DAC (E_{DAC}) and the deglitch filter (E_{LPF}) further degrade the output spectrum. This section analyzes various noise sources in DDS. Various spur-reduction techniques have been proposed [1–4], of which $\Sigma\Delta$ noise shaping has attracted attention recently. We will introduce modified DDS architectures with $\Sigma\Delta$ noise shaping in Section 10.4.

10.3.1 Phase Noise Due to Clock Jitter

A DDS functions like a frequency divider with a reference sampling clock as its input and a divided frequency as its output provided by the DAC. The spectral purity of the reference clock is the major contributor to the DDS output phase noise floor, even though its effect is reduced during the frequency division process. As in the phase noise analysis of a frequency divider given in Chapter 3, the clock will be assumed to have a sinusoidal FM signal described by $A\cos[\omega_s t + \beta\sin(\omega_j t)]$, where ω_s is the DDS sample clock frequency, ω_j is the FM jitter signal frequency, and β is the modulation index. The DDS output SNR due to the jittered clock is derived as [5]:

$$\text{SNR} \approx 3.01 - 20\log\beta + 20\log\left(\frac{f_s}{f_o}\right)\ \text{dB} \tag{10.6}$$

where $3.01 - 20\log\beta$ is equivalent to the SNR for an FM signal. Therefore, the phase noise of the DDS output is improved over its clock phase noise by an amount of $20\log(f_s/f_o) = 20\log(2\cdot\text{OSR})$ dB, where the OSR is defined as $f_s/2f_o$. For example, the phase noise difference between two output frequencies of 10 MHz and 100 MHz should be $20\log(10) = 20$ dB if the same clock frequency is used to generate the two frequencies. DDS with a high oversampling ratio can hence achieve lower phase noise. Additional spur attenuation is possible if clock buffers with fast transitions are employed since a high-slew-rate reference clock spends less time in the transition region where noise can cause jitter. Such clock buffers work as limiters or squaring circuits. The limiting process converts a clock with a sinusoidal waveform to a square waveform and, thus, reduces the AM spurs. Note that AM noise is converted to PM noise that directly affects the DDS output spectrum. By limiting the AM noise, the clock buffers provide additional attenuation of the clock jitter and, thus, improve the DDS output spectral purity. Although a sharp transition clock buffer is difficult to implement at high clock frequencies, it is good practice to maintain balanced clock tree networks with sufficient strength to drive the flip-flops and current switches in the DDS. In addition, it is important to minimize coupling from digital switching noise and power-supply noise to the high-frequency clock and analog signals.

10.3.2 Spurs Due to Discrete Phase Accumulation

For an N-bit phase accumulator, the desired output period is given by

$$T_o = \frac{2^N}{\text{FCW}} \cdot T_{\text{clk}} \tag{10.7}$$

In addition, there is another periodicity in the discrete phase-accumulation process that generates spurious tones in the frequency domain. The spur period can be determined by

$$T_{\text{spur}} = \frac{2^N}{\text{GCD}(\text{FCW}, 2^N)} \cdot T_{\text{clk}} \tag{10.8}$$

where the *greatest common divisor* (GCD) of a and b is denoted by GCD(a, b). Note that the spur period is an integer multiple of the desired output period since the FCW must be an integer multiple of GCD(FCW, 2^N). In general, the spurs of an N-bit phase accumulation are equally spaced and located at frequencies of

$$f_{\text{spur}} = F_n \cdot \frac{\text{GCD}(\text{FCW}, 2^N)}{2^N} \cdot f_{\text{clk}} \tag{10.9}$$

where F_n is an integer number used to number the spurs sequentially. Obviously, the desired output frequency of f_o has been included in the above expression. The number of spurs between the harmonics of the output fundamental tones is hence given by

$$k_o = \frac{T_{\text{spur}}}{T_o} - 1 = \frac{\text{FCW}}{\text{GCD}(\text{FCW}, 2^N)} - 1 \tag{10.10}$$

Within the entire sampling bandwidth (i.e., from $-f_{\text{clk}}/2$ to $f_{\text{clk}}/2$), the total number of spurs due to discrete phase accumulation is given by

$$k_{\text{total}} = \frac{T_{\text{spur}}}{T_{\text{clk}}} - 1 = \frac{2^N}{\text{GCD}(\text{FCW}, 2^N)} - 1 \tag{10.11}$$

where the harmonics of the fundamental tone have been included, and the fundamental tone itself has been excluded. Considering only the positive frequency band, the number of spurs in the Nyqusit band from 0 to $f_{\text{clk}}/2$ is given by

$$k_{\text{Nyquist}} = \frac{2^{N-1}}{\text{GCD}(\text{FCW}, 2^N)} \tag{10.12}$$

When the input frequency word is a power of two, that is, FCW = 2^i, there will be no spurs due to phase accumulation. In this case, the accumulator repeats at the same value after every overflow. As a result, the spurs overlap with the harmonics of the fundamental tone; that is

$$f_{\text{spur1}} = \frac{\text{GCD}(\text{FCW}, 2^N) \cdot f_{\text{clk}}}{2^N} = \frac{\text{FCW} \cdot f_{\text{clk}}}{2^N} = f_o \tag{10.13}$$

Example 10.1 Phase Accumulator Spurs
Analyze the spurious tone locations for an accumulator with length N equal to 8, and FCW equal to (a) 96 and (b) 64.

Solution: (a) When FCW = 96, the spur period is given by

$$T_{\text{spur}} = \frac{2^N}{\text{GCD}(\text{FCW}, 2^N)} = \frac{256}{\text{GCD}(96, 256)} = \frac{256}{32} = 8$$

Assume that the accumulator starts with a value of zero. With input of 96, the accumulator values at each time step are given as follows: 0, 96, 192, 32, 128, 224, 64, 160, 0, 96, . . . , which shows that the accumulator value indeed repeats every eight clock cycles. The desired output frequency is given by

$$f_o = \frac{\text{FCW} \cdot f_{\text{clk}}}{2^N} = \frac{96}{256} \cdot f_{\text{clk}} = 0.375 \cdot f_{\text{clk}}$$

This corresponds to the overflow of the accumulator every 256/96 clock cycles. In addition, a secondary periodicity repeats every eight clock cycles, that is, at multiples of

$$f_{\text{spur}} = \frac{\text{GCD}(\text{FCW}, 2^N) \cdot f_{\text{clk}}}{2^N} = \frac{f_{\text{clk}}}{8} = 0.125 \cdot f_{\text{clk}}$$

The spurious tones are hence at the multiple of ±1/3 of the fundamental frequency f_o. In the example, there are 96/32 – 1 = 2 spurs between the harmonics of the output fundamental, and there are a total of 128/32 = 4 spurs in the Nyquist band from 0 to $f_{\text{clk}}/2$. The four spurs are located at $f_{\text{clk}}/8$, $f_{\text{clk}}/4$, $3f_{\text{clk}}/8$, and $f_{\text{clk}}/2$.

(b) When FCW = 64, without losing generality, assume the accumulator initial value is zero. The accumulator values at each time step are given as follows: 0, 64, 128, 192, 0, 64, . . . , which means that the accumulator repeats the same value after every overflow. Thus, there is no secondary periodicity except the desired output period of $4T_{\text{clk}}$. Therefore; no spurs will be seen in the output spectrum.

10.3.3 Spurs and Quantization Noise Due to Phase Truncation

While a pure sinusoidal waveform is desired at the DDS output, spurious tones can occur mainly due to a number of nonlinear processes. In order to reduce the lookup table's ROM size, the phase word is normally truncated before being used as the ROM address. This truncation process introduces quantization noise, which can be modeled as a linear additive noise to the phase of the sinusoidal wave. The ROM word length is normally limited by the finite number of bits of the available DAC.

First, the phase truncation will be analyzed in the time domain. At time step n, the N-bit phase word at the output of the N-bit phase accumulator is updated as [6]

$$\Phi[n+1] = (\Phi[n] + \text{FCW}) \bmod 2^N \tag{10.14}$$

where $\Phi[n]$ represents the phase at time step n, and A mod B represents taking the integer residue of A modulo B. For example, 26 mod 16 = 10. To reduce the ROM size, only the P most significant bits of the accumulator output are used to address the lookup table. Truncating the N-bit phase word into P-bits causes a truncation error E_p, expressed as

$$E_p[n+1] = (E_p[n] + R) \bmod 2^{N-P} \tag{10.15}$$

where R is the least significant $(N-P)$ bits of the FCW value given by

$$R = \text{FCW} - \left\lfloor \frac{\text{FCW}}{2^{N-P}} \right\rfloor \times 2^{N-P} \tag{10.16}$$

where $\lfloor\ \rfloor$ denotes the truncation to keep the integer part. Hence, the output amplitude of the NCO can be expressed as

$$\begin{aligned} S[n] &= \sin\left(\frac{2\pi(\Phi[n] - E_p[n])}{2^N}\right) \qquad (10.17)\\ &= \sin\left(\frac{2\pi\Phi[n]}{2^N}\right)\cos\left(\frac{2\pi E_p[n]}{2^N}\right) - \cos\left(\frac{2\pi\Phi[n]}{2^N}\right)\sin\left(\frac{2\pi E_p[n]}{2^N}\right) \end{aligned}$$

where $S[n]$ is the amplitude at time step n. This can be compared to the ideal sinusoidal waveform $s(t)$ given by (10.5). For small truncation error, the above equation becomes

$$S[n] \approx \sin\left(\frac{2\pi\Phi[n]}{2^N}\right) - \frac{2\pi E_p[n]}{2^N} \cdot \cos\left(\frac{2\pi\Phi[n]}{2^N}\right) \tag{10.18}$$

The first term gives the desired sinusoidal output, and the second term is the error introduced by phase truncation. As shown, the phase-truncation error gives an AM term on the quadrature output. The phase-error sequence represented by the truncated $N - P$ bits satisfies the condition that $|E_p[n]| < 2^{N-P}$. The phase truncation causes errors only when $\text{GCD}(\text{FCW}, 2^N) < 2^{N-P}$. Otherwise, the $N - P$ LSBs of the phase word vanish, and the phase truncation does not cause any error. For example, if $N = 8$, $P = 6$, and $\text{FCW} = 2^2 =$ 3'b100, where 3'b100 denotes the decimal number 4 in 3-bit Verilog binary format, $\text{GCD}(4, 2^8) = 2^2 = 2^{8-6}$, and the two LSBs of the phase word are always equal to zero. Thus, phase truncation of the two LSBs does not cause any error.

Phase-truncation error is periodic, and its periodicity can be understood intuitively by analyzing (10.15). It is evident that $E_p[n]$ can be modeled by an $(N - P)$-bit small accumulator with R as its input. The accumulator is called small compared to the N-bit accumulator without truncation. Hence, the period of the error sequence $E_p[n]$ is given by $2^{N-P}/R$. Thereby, the periodic truncation error creates major spurs at the harmonic frequencies of $f_{\text{clk}} R/2^{N-P}$. Figure 10.5 shows the periodic sawtooth waveform of the phase-truncation error. Note that the slope of the sawtooth is given by

$$\text{Slope} = \frac{R}{T_{\text{clk}}} = R \cdot f_{\text{clk}} \tag{10.19}$$

and the frequency of the sawtooth waveform is

$$f_{\text{saw}} = \frac{\text{slope}}{2^{N-P}} = \frac{R \cdot f_{\text{clk}}}{2^{N-P}} \tag{10.20}$$

which is the output frequency of an $(N - P)$-bit accumulator with R as the input.

From (10.9), the $(N - P)$-bit accumulator with R as the input also causes spurs at the frequencies of

$$f_{\text{spur}} = F_n \cdot \frac{\text{GCD}(R, 2^{N-P})}{2^{N-P}} \cdot f_{\text{clk}} \tag{10.21}$$

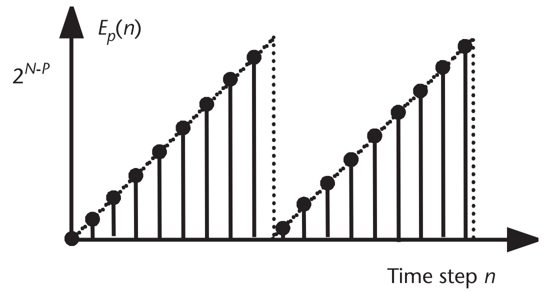


Figure 10.5 Quantization error sequence of phase truncation.

This expression covers the major spur frequencies $F_n \cdot f_{\text{clk}} R/2^{N-P}$ as well. Note that the least significant $(N-P)$-bit frequency word R can be replaced with the original N-bit FCW since the P most significant bits of the FCW do not affect the operation of the small accumulator with a length of $N-P$ bits. Therefore, the phase-truncation spurs are mixed with the DDS output frequency, generating spurs at offset frequencies of

$$f_{\text{spur}} = F_n \cdot \frac{\text{GCD}(\text{FCW}, 2^{N-P})}{2^{N-P}} \cdot f_{\text{clk}} \tag{10.22}$$

where F_n is an integer to number the spurs sequentially. Note that the spurs due to the $N-P$ bit accumulator form a subset of the spurs due to the N-bit accumulator given in (10.9). According to (10.12), the number of spectral lines in the Nyquist band due to the truncated small accumulator with length of $N-P$ bits is given by

$$\Lambda = \frac{2^{N-P-1}}{\text{GCD}(\text{FCW}, 2^{N-P})} \tag{10.23}$$

It should be pointed out that the above expression considers all the potential spur locations due to phase truncation. Those spurs have different magnitudes and some of them may even vanish. A more complete analysis of spur locations and magnitude involves a Fourier transformation. If the sawtooth waveform shown in Figure 10.5 can be expressed as a Fourier series, the spurs associated with phase-truncation errors can be analyzed. However, the sawtooth waveform is defined to have a value of zero at the discontinuity, which violates the Dirichlet condition required for a Fourier transform. The Dirichlet condition requires that the function take its average value, which is 2^{N-P} in this case, at a point of discontinuity. B. Kim, H. T. Nicholas, and H. Samueli [7–9] represent the phase error as the superposition of an ideal Dirichlet sawtooth and a correcting waveform such that both waveforms satisfy the Dirichlet conditions, and the sampled, superposed waveform equals the actual error sequence. Without going into detail, we express the error sequence as [7]

$$E_p[n] = \frac{-2^{N-P}}{2\Lambda} \sum_{K=1}^{\Lambda} \left[\cot\left(\frac{K\pi}{2\Lambda}\right) \sin\left(2\pi K \frac{\text{FCW}}{2^{N-P}} \cdot n\right) - \cos\left(2\pi K \frac{\text{FCW}}{2^{N-P}} \cdot n\right) \right] \tag{10.24}$$

Note that the error sequence is expressed as a sum of Λ distinct spurious tones, where Λ is defined in (10.23). Combining the sine and cosine terms into a complex Fourier series yields

$$E_p[n] = \sum_{K=1}^{\Lambda} \left\{ \xi_K \exp\left(j2\pi K \frac{\text{FCW}}{2^{N-P}} \cdot n\right) \cdot \exp[j\Psi(K, \Lambda)] \right\} \tag{10.25}$$

where the complex phasor has the following magnitude and angle:

$$\text{Magnitude: } \xi_K = \frac{2^{N-P}}{2\Lambda} \operatorname{cosec}\left(\frac{K\pi}{2\Lambda}\right) \tag{10.26}$$

$$\text{Angle: } \Psi(K, \Lambda) = -\cot\left(\frac{K\pi}{2\Lambda}\right)$$

Now, applying the obtained error sequence E_p into (10.18), the NCO output with the phase-truncation error can be expressed as

$$S[n] \approx \sin\left(2\pi \frac{\text{FCW}}{2^N} \cdot n\right) - \frac{2\pi}{2^N} \cdot \sum_{K=1}^{\Lambda} \xi_K$$
$$\cdot \left\{ \exp\left[j2\pi \left(\frac{\text{FCW}}{2^N} + K\frac{\text{FCW}}{2^{N-P}} \right) \cdot n \right] + \exp\left[-j2\pi \left(\frac{\text{FCW}}{2^N} - K\frac{\text{FCW}}{2^{N-P}} \right) \cdot n \right] \right\} e^{j\Psi(K,\Lambda)} \tag{10.27}$$

Note that the first term represents the desired output sinusoidal waveform, and the second term is the summation of the Λ uniformly spaced phase-truncation spurs with magnitude of

$$\xi_{K\pm} = \frac{2\pi}{2^N} \xi_K = \frac{\pi}{2^{P+1}\Lambda} \operatorname{cosec}\left(\frac{K\pi}{2\Lambda}\right) \tag{10.28}$$

and located at frequencies given in (10.9). Next, we define a variable

$$\Gamma = \frac{\text{FCW}}{\text{GCD}(\text{FCW}, 2^{N-P})} \tag{10.29}$$

so that the frequency FCW/2^{N-P} can be expressed as a rational fraction of two prime integers. As shown in (10.12), an accumulator with length N may cause 2^{N-1}/GCD(FCW, 2^N) potential spurs uniformly distributed in the Nyquist band from 0 to $f_{\text{clk}}/2$. The spurs can be sequentially numbered with the frequency number F_n defined in (10.9). The kth spur index can be found using the following rules:

- If neither $F_n - \Lambda$ nor $-F_n - \Lambda$ is divisible by two, then the magnitude of the spur at F_n is zero.
- If $F_n - \Lambda$ is divisible by two, then the kth spur index is given by

$$K = \left(\frac{F_n - \Lambda}{2^P} \cdot \Gamma^{\Lambda - 1} \right) \bmod (2\Lambda) \tag{10.30}$$

- If $-F_n - \Lambda$ is divisible by two, then the kth spur index is given by

$$K = \left(\frac{-F_n - \Lambda}{2^P} \cdot \Gamma^{\Lambda - 1} \right) \bmod (2\Lambda) \tag{10.31}$$

According to the above Fourier analysis on the sawtooth error sequence shown in Figure 10.5, the *k*th spur magnitude due to phase-truncation error can be summarized as [7]

$$\xi_{K\pm} = \begin{cases} \dfrac{\pi}{2^{P+1}\Lambda}\operatorname{cosec}\left(\dfrac{K\pi}{2\Lambda}\right) & \text{if } \pm F_n - \Lambda \text{ is divisible by 2} \\ 0 & \text{otherwise} \end{cases} \tag{10.32}$$

The above spur magnitude is a monotonically decaying function when K increases. The worst-case spur magnitude normalized to a signal magnitude of unity can thus be obtained by setting $K = 1$:

$$\begin{aligned}\xi_{\pm 1\,\text{Worst spur}} &= \frac{\dfrac{\pi \text{GCD}(\text{FCW}, 2^{N-P})}{2^{N-P}}}{2^P \sin\left[\dfrac{\pi \text{GCD}(\text{FCW}, 2^{N-P})}{2^{N-P}}\right]} \\ &= 2^{-P}\,\text{sinc}^{-1}\left[\frac{\pi \text{GCD}(\text{FCW}, 2^{N-P})}{2^{N-P}}\right]\end{aligned} \tag{10.33}$$

It is important to note that the spurs due to phase truncation are nonexistent when the least significant $(N - P)$ bits of the FCW value are zeros, that is, $\text{GCD}(\text{FCW}, 2^{N-P}) = 2^{N-P}$. However, for $\text{GCD}(\text{FCW}, 2^{N-P}) < 2^{N-P}$, the magnitude of the worst spurs is a decaying function of $2^{N-P}/\text{GCD}(\text{FCW}, 2^{N-P})$ with the maximum value of $(\pi/2) \cdot 2^{-P} = -6.02 \cdot P + 3.922$ dB when $\text{GCD}(\text{FCW}, 2^{N-P}) = 2^{N-P-1}$, which means there is only one spur $(\Lambda = 1)$. If the number of spurs is large, meaning that $\text{GCD}(\text{FCW}, 2^{N-P}) \ll 2^{N-P}$, the worst spur magnitude asymptotically approaches the lower bound of 2^{-P}, or $-6.02 \cdot P$, in decibels. An example of where spur magnitude approaches the lower bound occurs when FCW and 2^{N-P} do not have common divisors greater than one. In this case, $\text{GCD}(\text{FCW}, 2^{N-P}) = 1$ and the number of discarded bits $(N - P)$ is large. In summary, the worst-case spur magnitude due to phase truncation can be estimated by

$$\begin{cases} \xi_{\max} = \dfrac{\pi}{2} \cdot 2^{-P} = -6.02 \cdot P + 3.92\ [\text{dBc}] & \text{if } \Lambda = 1\ [\text{i.e., GCD}(\text{FCW}, 2^{N-P}) = 2^{N-P-1}] \\ \xi_{\min} = 2^{-P} = -6.02 \cdot P\ [\text{dBc}] & \text{if } \Lambda = \infty\ [\text{i.e., GCD}(\text{FCW}, 2^{N-P}) \ll 2^{N-P}] \\ \xi = 0 & \text{if GCD}(\text{FCW}, 2^{N-P}) = 2^{N-P} \end{cases} \tag{10.34}$$

Therefore, the worst spur magnitude can be reduced by 3.922 dB if we force $\text{GCD}(\text{FCW}, 2^{N-P}) = 1$, by adjusting the FCW to be relatively prime to 2^{N-P}. A modification of the phase accumulator was proposed in [7]. The modification causes the output of the phase accumulator to behave as if the accumulator word length is $N + 1$, with the LSB to be 1. Thus, a relatively coprime FCW and 2^{N-P} can result in $\mathrm{GCD}(2 \cdot \mathrm{FCW} + 1, 2^{N-P+1}) = 1$.

The periodic phase-truncation error is equivalent to a nonuniform sampling process [10], in which the sinusoidal waveform is sampled nonuniformly with sampling advancement offsets, which means sampling takes place earlier than the DDS clock edge. Summing all the spur energy gives the total noise power. The noise-to-signal ratio (NSR) due to phase truncation can be found as [11, 12]

$$\mathrm{NSR} = \left\{ \frac{\mathrm{sinc}\left[\frac{\pi}{2^N} \mathrm{GCD}(\mathrm{FCW}, 2^{N-P})\right]}{\mathrm{sinc}\left(\frac{\pi}{2^P}\right)} \right\}^2 - 1 \tag{10.35}$$

It is interesting to mention the following properties of the phase-truncation noise:

1. If

$$2\Lambda = \frac{2^{N-P}}{\mathrm{GCD}(\mathrm{FCW}, 2^{N-P})} = 1$$

   the least significant $(N - P)$ bits of the FCW are zeros, and there are no spurs due to phase truncation. The only spectral line in the range of $-f_{\mathrm{clk}}/2 \le f \le f_{\mathrm{clk}}/2$ is the desired signal.
2. For a fixed number of phase bits P, the NSR is an increasing function of the number of spectral lines Λ, which results in a higher noise level.
3. For a fixed number of spectral lines Λ, the NSR is a decreasing function of the number of phase bits, P. Therefore, more phase bits lead to a lower noise level.

Therefore, the upper and lower bounds of the phase-truncation noise can be found by assuming an infinite number of spurs [$\Lambda = \infty$, i.e., $\mathrm{GCD}(\mathrm{FCW}, 2^{N-P}) \ll 2^{N-P}$ according to (10.23)] and one spur [$\Lambda = 1$, i.e., $\mathrm{GCD}(\mathrm{FCW}, 2^{N-P}) = 2^{N-P-1}$] as follows:

$$\begin{cases} \mathrm{NSR}_{\max} = \mathrm{sinc}^{-2}\left(\frac{\pi}{2^P}\right) - 1 & \text{if } \Lambda = \infty \\ \mathrm{NSR}_{\min} = \tan^2\left(\frac{\pi}{2^{P+1}}\right) & \text{if } \Lambda = 1 \\ \mathrm{NSR} = 0 & \text{if } \mathrm{GCD}(\mathrm{FCW}, 2^{N-P}) = 2^{N-P} \end{cases} \tag{10.36}$$

For a large number of phase bits, P, and considering trigonometric approximations, the above equations can be approximated as

$$\begin{cases} \text{NSR}_{\max} \approx \dfrac{1}{3}\left(\dfrac{\pi}{2^P}\right)^2 = -6.02 \cdot P + 5.17 \text{ [dB]} & \text{if } \text{GCD}(\text{FCW}, 2^{N-P}) \ll 2^{N-P} \\ \text{NSR}_{\min} = \dfrac{\pi}{2}\left(\dfrac{1}{2^P}\right)^2 = -6.02 \cdot P + 3.92 \text{ [dB]} & \text{if } \text{GCD}(\text{FCW}, 2^{N-P}) = 2^{N-P-1} \\ \text{NSR} = 0 & \text{if } \text{GCD}(\text{FCW}, 2^{N-P}) = 2^{N-P} \end{cases} \tag{10.37}$$

Comparing to the spur magnitude given in (10.34), we can see the following trends: (1) if there is only one spur, the spur power reaches the maximum, and the resultant NSR is minimum; (2) however, with an infinite number of spurs, the spur power reaches the minimum, and the resultant NSR is maximum.

10.3.4 Quantization Noise Due to Finite Number of Amplitude Bits

Amplitude quantization errors occur when storing the sinusoidal values in the lookup table. The lookup takes in a fixed number of phase bits and converts them to the equivalent sine amplitude word with a finite number of bits, which is normally chosen based on the available number of DAC input bits. Additional noise will be introduced when compression algorithms are used for ROM size reduction. ROM compression is achieved by storing only part of the sinusoidal values and reconstructing other data points by interpolation. Adding the amplitude error due to finite amplitude resolution E_A and the amplitude error due to ROM compression E_{COM}, the NCO output given in (10.18) can be rewritten as

$$S[n] \approx \sin\left(\frac{2\pi\Phi[n]}{2^N}\right) - \frac{2\pi E_p[n]}{2^N} \cdot \cos\left(\frac{2\pi\Phi[n]}{2^N}\right) + E_A + E_{\text{COM}} \tag{10.38}$$

A quick look at the sinusoidal function $\sin(x) \approx x - x^3/6$ reveals that fewer bits are required to represent the amplitude of a sine wave than the number of its phase bits since the amplitude word of $\sin(x)$ is less than the phase word x in the first quadrant. Hence, more phase bits than amplitude bits should be used for reducing the spurs.

As shown, the discrete phase-accumulation process with finite phase bits generates spurious tones that are worsened by the phase truncation. Unlike phase truncation, the effect of finite amplitude word length generates random quantization noise. As discussed in Chapter 9, at the Nyquist rate, the SNR due to the quantization noise power that falls into the signal band is given by $3/2 \times 2^{2D} = 6.02D + 1.76$ dB, where D is the finite amplitude word length. Considering the quantization noise due to both the finite phase bits (10.34) and the finite amplitude bits, the worst-case spur magnitude at the DDS output is given by

$$\xi_{\max} \text{ [dBc]} = 10 \log\left(\frac{\pi^2}{4} \cdot 2^{-2P} + \frac{2}{3} \cdot 2^{-2D}\right) \tag{10.39}$$

The above spur estimation is plotted in Figure 10.6. Note that the phase-truncation error causes peak output spurs that are generally 3.92 dB + 1.76 dB = 5.68 dB above the quantization noise floor due to the finite amplitude bit effect. Therefore, we need to choose $P = D + 1$ for DDS designs such that both the phase spurs and the amplitude noise floor are reached at about the same quantization noise level. Practically, $P \geq D + 2$ should be chosen to ensure that the DDS output spur magnitude is dominated by the finite number of amplitude bits and not by the finite number of phase bits since it is much more costly to increase the number of DAC bits than it is to increase the number of phase bits.

10.3.5 DAC Nonlinearities and Aliased Images

Since it is easier to minimize the quantization noise generated in an NCO than it is to optimize the DAC analog circuits, the normal design practice is to use more phase bits than amplitude bits in the lookup table. Therefore, the DAC nonlinearity, distortion, and quantization noise become the dominant factors in determining the DDS output spectrum purity. For a first-order approximation, the number of DAC input bits determines the DDS broadband signal-to-spurious ratio, which is roughly 6 dB per bit. For example, a 12-bit, digitized, sinusoidal output will theoretically provide a SNR of 72 dB. This SNR calculation models an ideal DAC with only quantization noise due to the finite input bits. The actual SNR depends on the quality of the DAC and the filter design, as well as the phase noise of the DDS clock frequency. Real DACs have nonlinearities due to process mismatches, imperfect bit-weight scaling circuits, nonideal switching characteristics, and so forth. The most

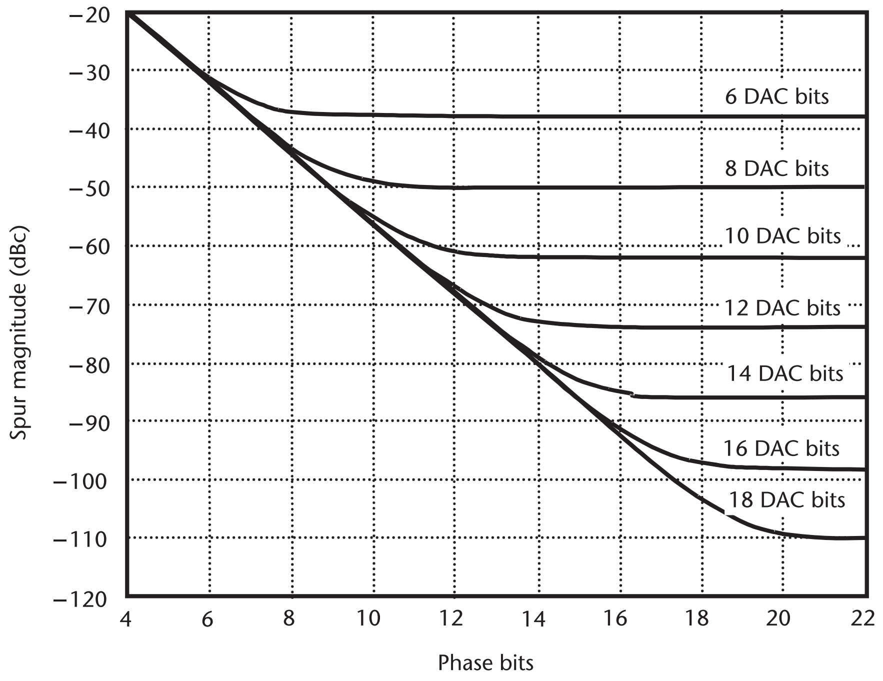


Figure 10.6 DDS worst-case spur magnitude due to finite phase bits and finite amplitude bits.

prominent DAC spurs are usually due to nonideal switching, manifested as lower-order harmonics of the fundamental, along with any nonlinearity in the DAC transfer function. Both quantization noise and the DAC nonlinearity produce a response that consists of spurs, which are harmonically related to the fundamental. Since the DAC is a sampled system, the DAC's sample-and-hold function imposes a *sinc* envelope to the DDS output spectrum. Theoretically, the DDS is able to generate frequencies from dc to the Nyquist frequency, which is half of the clock frequency. However, the implementation of the deglitch filter limits the practical upper bound of the DDS output frequency to about 40% of the clock frequency. (Note that generating outputs at the Nyquist rate requires an infinitely fast roll off in the deglitch filter to reject the aliased images.) In addition, the clock frequency, the DDS output frequency, and their harmonics tend to mix with each other and alias back to the Nyquist band. It is possible to predict the spur locations using a well-defined mathematical model. The frequencies of the discrete spurs and their amplitude are dependent on the ratio of the generated frequency f_o to the sampling clock frequency f_s. As a nonlinear device, the DAC creates discrete aliased images at the following frequencies:

$$f_{\text{image}} = m \cdot f_s \pm n \cdot f_o \tag{10.40}$$

where m and n are integers. Those images will be attenuated by the deglitch filter transfer function and the DAC's sample-and-hold function, $\text{sinc}[\pi \cdot f_o/f_s]$. Assuming that there are no spurs caused by finite phase and amplitude bits, the DDS output spectrum with aliased images tones is shown in Figure 10.7.

Significant benefits arise in DDS applications from setting the DDS output frequency at a subharmonic of the sampling clock frequency, which corresponds to setting the input FCW to be a power of two. Hence, the DDS operates as a frequency divider, and all the images are phase coherent with the clock, and no additional jitter is produced. Careful selection of the clock frequency can eliminate nth-order aliased images. It is necessary to consider only signal images beating

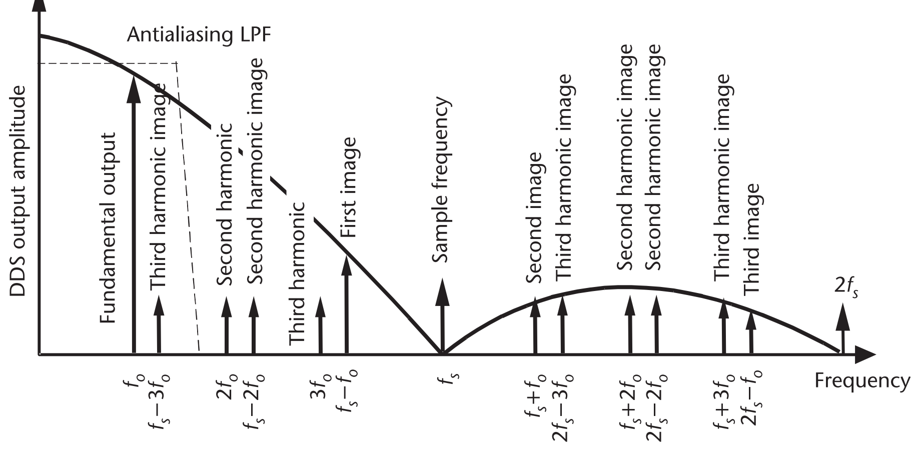


Figure 10.7 DDS output spectrum with aliased image tones.

with the fundamental of the clock, and it is possible to ignore higher-order clock harmonics and higher-order cross products since they are weaker in magnitude. The condition to eliminate the nth-order images along with all lower-order aliased images can be shown to be $f_{\text{clk}} > (n+1)f_{\text{max}}$, where n is the order of aliased images and f_{max} is the upper edge of the band of interest. For example, if the DDS maximum output frequency f_{max} is 25 MHz, a clock frequency of $f_{\text{clk}} > 125$ MHz will eliminate all aliased images up to the fourth order.

10.3.6 Oversampling Effect

As discussed in Chapter 9, operating a sampled system at a frequency higher than the Nyquist rate reduces the quantization noise density due to the increased bandwidth. Oversampling effects can also be applied to DDS, since it is intrinsically a sampled system, before data conversion. Although the random quantization noise floor can be lowered, oversampling does not affect the spur level since the spurs occur at deterministic frequencies and are therefore not randomized noise. Assuming the DAC's effective number of bits D dominates the DDS quantization noise, the DDS signal-to-noise spectral density can be represented by

$$\text{SNR} = 6.02 \cdot D + 1.76 + 3 \log_2 \text{OSR [dB]} \tag{10.41}$$

where the oversampling ratio is defined as

$$\text{OSR} = \frac{f_{\text{clk}}}{2f_{\text{out}}} = \frac{2^N}{2 \cdot \text{FCW}} = \frac{2^{N-1}}{\text{FCW}} \tag{10.42}$$

Hence, every doubling of the OSR results in a 3-dB improvement of the quantization noise SNR, which is equivalent to an increase of half of an input bit. Although the deterministic spurs are not affected by the oversampling effect, the DDS output spectral purity is limited by the available circuit speed. When the DDS clock is close to its maximum operating frequency, the spurious free dynamic range (SFDR) of the DDS output will get worse as the clock frequency increases. Thus, at high clock frequencies, the quantization noise cannot be effectively reduced by as much as would be predicted by oversampling theory.

10.4 ΣΔ Noise Shaping in DDS

It was shown that the phase-truncation process associated with the conventional DDS architecture introduces quantization error and spurs. The DDS output due to the quantization errors of phase truncation E_p and finite amplitude resolution E_A is given in (10.38). It has been shown that the phase error is amplitude modulated on the quadrature signal with respect to the desired signal output. To avoid aliasing during data conversion, the synthesized frequency must be smaller than the DDS clock frequency. Thus, oversampling is always encountered in DDS, allowing noise-shaping techniques to be used to shift the phase quantization error to a higher frequency band, where the noise can eventually be removed by the deglitch filter after the DAC. ΣΔ noise shaping has been employed in DDS designs [6, 13, 14]. ΣΔ modulation can be implemented in both the frequency and phase domains in a DDS. The frequency domain ΣΔ modulation has the advantages of increased dynamic range due to constant input and reduced accumulator size due to FCW truncation in the frequency domain. Noise-shaping techniques can be used either to increase the DDS resolution for high-performance applications or to reduce the ROM size for low-cost applications. Using the ΣΔ interpolator to remove the phase-truncation error, we can build a larger accumulator (e.g., $n > 32$ bits) to achieve finer resolution with low quantization noise. Alternatively, without degrading the output spectral purity, we can truncate even more phase bits to address a much smaller ROM size [14].

10.4.1 DDS Using Phase Domain ΣΔ Noise Shaping

Recall that ΣΔ modulators discussed in Chapter 9 can noise-shape the quantization error towards higher frequency bands. In a DDS, various ΣΔ topologies can also be used to reduce the phase-truncation errors. Figure 10.8 illustrates the implementation of phase domain noise shaping using a feedback ΣΔ modulator, as shown in Figure 10.8(a), or using a feedforward ΣΔ modulator, as shown in Figure 10.8(b). As shown in Figure 10.8(b), a *k*th-order ΣΔ noise shaper with a unique transfer function can be added after the phase truncation. For noise shaping, the discarded

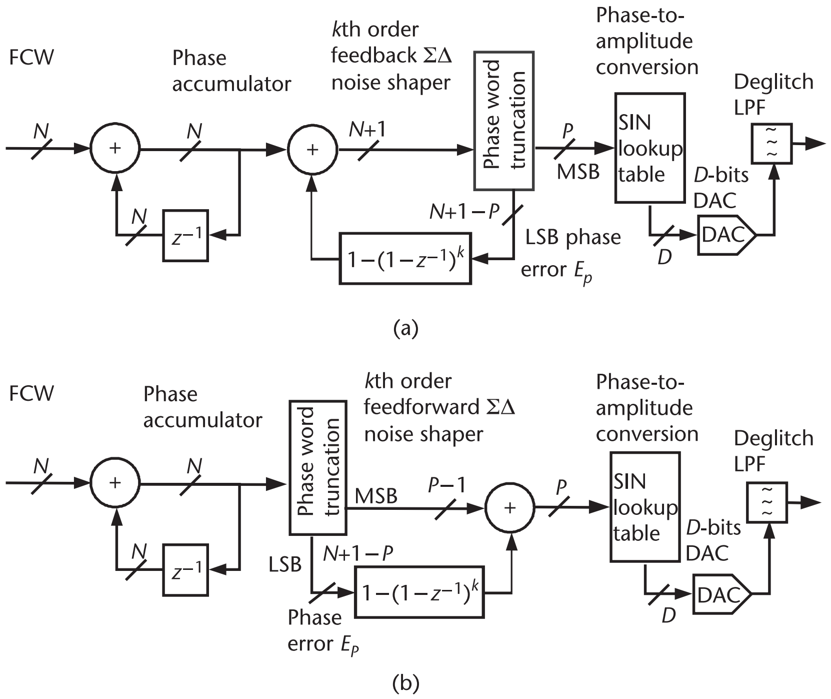


Figure 10.8 DDS architecture with a *k*th-order, phase domain ΣΔ noise shaper to reduce phase-truncation error: (a) DDS with phase domain feedback ΣΔ noise shaper, and (b) DDS with phase domain feedforward ΣΔ noise shaper.

$(N + 1 - P)$ LSB of the phase word, E_p, is fed into a kth-order $\Sigma\Delta$ noise shaper with transfer function of $1 - (1 - z^{-1})^k$. For both phase domain feedback and feedforward $\Sigma\Delta$ noise shapers with a noise transfer function of $1 - (1 - z^{-1})^k$, the resulting DDS output can be expressed as

$$S[n] = \sin\left\{\frac{2\pi[\Phi[n] - E_p[n] \cdot (1 - z^{-1})^k]}{2^N}\right\} \tag{10.43}$$

$$\approx \sin\left(\frac{2\pi\Phi[n]}{2^N}\right) - \frac{2\pi E_p[n]}{2^N} \cdot (1 - z^{-1})^k \cdot \cos\left(\frac{2\pi\Phi[n]}{2^N}\right)$$

It can be seen that the phase error E_p is highpass filtered by the $\Sigma\Delta$ interpolator before the phase-to-amplitude conversion via the lookup table. This highpass noise shaping greatly reduces the close-in phase noise and decorrelates the phase-truncation errors. Thus, the spurious components at the DDS output are greatly reduced or eliminated. Note that the feedback $\Sigma\Delta$ noise shaper requires an $(N + 1)$-bit adder, while the feedforward $\Sigma\Delta$ noise shaper requires only a P-bit adder.

To demonstrate the $\Sigma\Delta$ shaping effect on the phase-truncation error, the modified DDS architecture with a fourth-order, phase domain, feedback $\Sigma\Delta$ noise shaping is simulated in MATLAB. The conventional DDS is also simulated as a comparison. We compare the output spectra after the phase truncation with a dc input to the accumulator in a conventional DDS (Figure 10.9) and a DDS with a fourth-order $\Sigma\Delta$ noise shaper (Figure 10.10). Figure 10.10 clearly demonstrates the highpass noise-shaping effect of the fourth-order $\Sigma\Delta$ interpolator with 80 dB/dec slope at the input of the ROM after the phase truncation. Note that the plot is not the DDS phase noise PSD but, instead, is the spectra of the digital phase words after the phase truncation. As discussed in Chapter 9, the noise-shaping slope of the phase noise PSD for a kth-order $\Sigma\Delta$ modulator is $20(k - 1)$ dB/dec. The fourth-order $\Sigma\Delta$ interpolator should provide a 60 dB/dec highpass noise-shaping slope for the phase noise PSD. The $\Sigma\Delta$ noise shaper moves close-in phase-truncation spurious components to a higher frequency band, where they can easily be removed by the deglitch lowpass filter.

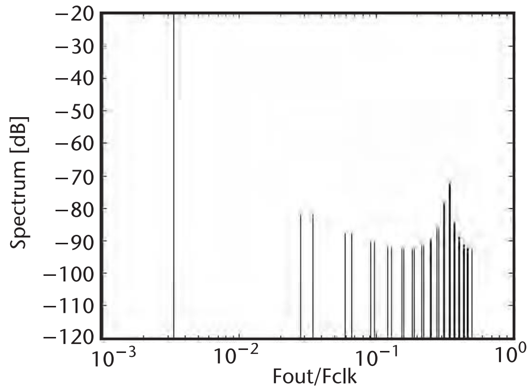


Figure 10.9 Output spectrum after the phase truncation for a conventional DDS.

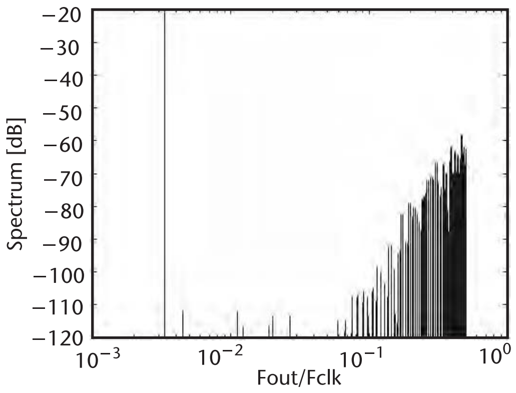


Figure 10.10 Output spectrum after the phase truncation for a DDS with a fourth-order, phase domain ΣΔ noise shaper.

10.4.2 DDS Using Frequency Domain ΣΔ Noise Shaping

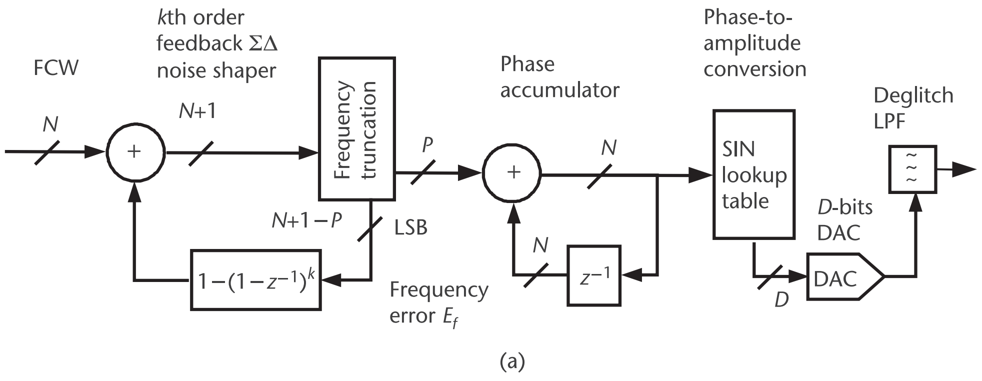
Similar to the phase domain ΣΔ noise shaping shown in Figure 10.8, Figure 10.11 illustrates frequency domain noise shaping using a ΣΔ modulator. Since the FCW in a DDS is normally a constant, the inputs to the first adder are constant as well, which benefits high-speed implementation using pipelined adders. Thus, the feedforward ΣΔ modulator gains an advantage of less hardware over the feedback ΣΔ modulator in the frequency domain since the first adder has only P bits in the feedforward ΣΔ topology. For frequency domain feedback and feedforward ΣΔ noise shapers with a noise transfer function of $1 - (1 - z^{-1})^k$, the resulting DDS output can be expressed as

$$S[n] = \sin\left\{2\pi n\left(\frac{\text{FCW}}{2^N} - \frac{E_f[n]\cdot(1 - z^{-1})^k}{2^P}\right)\right\} \tag{10.44}$$

It can be seen that the frequency error E_f is highpass filtered by the ΣΔ interpolator before the phase accumulation. Note that the truncated frequency error will be accumulated in the phase domain, which is the drawback of the frequency domain ΣΔ noise shaping. Truncating the FCW also reduces the size of the accumulator, which is often the speed bottleneck in a high-speed DDS. If the FCW is truncated to P bits, there is no need to perform the phase truncation after the accumulator. Note that frequency word truncation will simply result in a loss of synthesis resolution should there be no ΣΔ interpolator that adds back the discarded FCW bits. The frequency domain ΣΔ noise shaping reduces the close-in phase noise as well. The noise-shaped spectrum will eventually be cleaned up by the deglitch filter. Ideal sinusoidal waveforms with greatly reduced close-in phase noise and spurious components can thus be achieved at the DDS output. It is of great commercial value to achieve very low-phase noise frequency synthesis through DDS due to its low cost and capability with digital CMOS integration.

10.4.3 ROM Size Reduction Using ΣΔ Noise Shaping

Without degrading the output SNR, ΣΔ noise shaping can also be used to reduce the ROM size, which often takes up the majority of the DDS area. In an N-bit



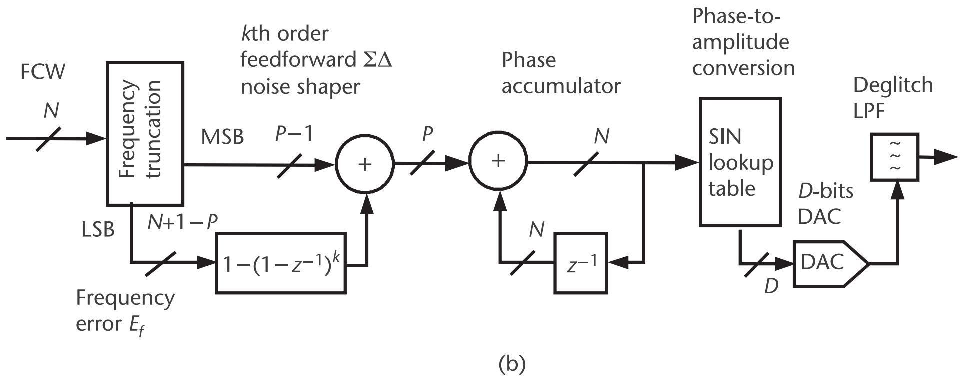


Figure 10.11 DDS architecture with a *k*th-order, frequency domain, ΣΔ noise shaper to reduce the frequency-truncation error: (a) DDS with frequency domain feedback ΣΔ noise shaper, and (b) DDS with frequency domain feedforward ΣΔ shaper.

sampled system, if the quantizer has $2N$ quantization levels equally spaced by Δ, then the maximum peak-to-peak amplitude is given by $v_{\max} = (2^N - 1)\Delta$. If the signal is sinusoidal, its power can be calculated as $P_s = 1/8 \cdot (2^N - 1)^2\Delta^2$. Chapter 9 shows that for an oversampled system with a kth-order ΣΔ modulator, the in-band rms quantization noise power is given by

$$P_N = \frac{\Delta^2}{12}\frac{\pi^{2k}}{2k+1}\left(\frac{1}{\mathrm{OSR}}\right)^{2k+1} \tag{10.45}$$

The SNR with oversampling and noise shaping can be found from the ratio of the signal power to the noise power given above. The performance improvement due to oversampling and ΣΔ noise shaping can be characterized by the number of effective bits, which leads to the same SNR over a fixed bandwidth:

$$\mathrm{SNR} = 6.02N + 1.76 + 3(2k+1)\log_2 \mathrm{OSR} - 10\log_{10}\left(\frac{\pi^{2k}}{2k+1}\right) \tag{10.46}$$

Therefore, the lowered quantization noise due to oversampling and noise shaping leads to an effectively increased number of quantizer bits by

$$N_{\text{effective}} \approx N + \frac{2k+1}{2} \log_2 \text{OSR} - 1.66 \cdot \log_{10}\left(\frac{\pi^{2k}}{2k+1}\right) \tag{10.47}$$

Note that the number of effective bits cannot be improved greatly without significant oversampling. At the Nyquist rate (OSR = 1), there is even a reduction in the number of effective bits due to the effect of the ΣΔ modulator. For OSR ≥ 4, we conclude that a kth-order ΣΔ noise shaping reduces the required number of phase bits at least by

$$P \approx P_{\text{required}} - 2k - 1 + 1.66 \cdot \log_{10}\left(\frac{\pi^{2k}}{2k+1}\right) \tag{10.48}$$

For instance, a fourth-order ΣΔ noise shaper will at least effectively reduce the required number of phase bits by four. To illustrate this point, we now describe a fully integrated DDS using a fourth-order phase domain ΣΔ modulator implemented in 0.35 μm CMOS technology with two poly and four metal layers [14]. A 16-bit accumulator is designed, and its 8 MSBs are used for addressing the lookup ROM. The 12-bit current-steering DAC is integrated to convert the NCO output to an analog signal. For 12-bit amplitude resolution in a conventional DDS without a ΣΔ modulator, at least 12 phase bits should be used, which requires a lookup ROM with $2^{12} \times 12$ bits. As the previous paragraph shows, the use of a fourth-order ΣΔ noise shaper effectively reduces the required number of phase bits by four. Thus, only 8 phase bits are used to address the ROM, which reduces the ROM size by a factor of 2^4, or 16 times, compared to that of a conventional DDS without a ΣΔ modulator. Notice that this reduction in ROM size is due only to the ΣΔ noise-shaping effect. A further ROM size reduction can be achieved using ROM compression algorithms. The total DDS core area is 1.11 mm^2. The 16-bit phase accumulator and the fourth-order ΣΔ modulator occupy a die area of 0.3 × 0.2 mm^2. The $2^8 \times 12$-bit ROM occupies only 0.3 × 0.3 mm^2, whereas the area would be 16 times larger without the ΣΔ noise shaper. In a conventional DDS, the ROM normally takes the majority of the die area, whereas the ROM takes only a small portion of the total area in this DDS implementation, which clearly demonstrates the advantage of using high-order ΣΔ noise shaping in DDS designs.

10.5 High-Speed ROM-Less DDS

DDS provides precise frequency resolution and direct-modulation capability. However, the majority of the DDSs designed so far have been limited to low-frequency applications with clock frequencies less than a few hundred megahertz. Digitally generating highly complex, wide-bandwidth waveforms at the highest possible frequency, instead of down near baseband, would considerably reduce the size, weight, and power requirements, as well as the cost, of a transmitter architecture.

The DDSs considered so far with linear DACs contain three major parts: an accumulator, a ROM for the sine lookup table, and a linear amplitude-to-amplitude DAC. In a ROM-less DDS, the ROM is removed, and a nonlinear DAC serves as both the digital sine wave generator and the phase-to-amplitude converter [15]. The sine-weighted DAC eliminates the sine lookup table, which is the speed and area bottleneck for high-speed DDS implementations.

This section discusses high-speed DDS designs suitable for multigigahertz clock frequencies and capable of synthesizing and modulating output frequencies as high as 5 GHz. A high-speed DDS includes a high-speed accumulator, column/row decoders, and a sine-weighted DAC. Sine-weighted, 8-bit DACs operating with clock frequencies from 2 to approximately 10 GHz have been implemented in indium phosphide (InP) [16], silicon germanium (SiGe) [17], and CMOS [18] technologies with spectral purities better than –45 dBc. In traditional DDSs, the ROM size is exponentially proportional to the desired phase resolution. The ROM occupies the majority of the DDS area and also limits its maximum operating frequency due to the delay through the multilayer decoders. Though many ROM compression methods have been proposed, such as trigonometric approximation and parabolic approximation [19], the problems indicated above still exist. An alternate approach is to replace the conventional linear DAC that converts digital amplitude words to an analog amplitude waveform with a nonlinear DAC that converts the digital phase word into an analog sine waveform directly [20].

Figure 10.12 shows the conceptual block diagram of the nonlinear DAC-based ROM-less DDS. The N-bit FCW feeds into a phase accumulator that controls the output frequency of a synthesized sine waveform. The output of the phase accumulator is truncated into W bits according to the SNR requirement of the sine output. The DDS architecture exploits the quadrant symmetry property of the sine function around $\pi/2$ and π. The two MSBs are used to determine in which sine wave quadrant the phase accumulator output resides, according to the quadrant symmetry of the sine wave. The remaining $W - 2$ bits are applied to the complementor and are used to generate the waveform of the sinusoid in the first quadrant.

The speed of the DDS is often limited by the speed of the phase accumulator, and, in turn, the speed of the accumulator depends upon the N-bit adder and the flip-flop design. The following sections discuss two architectures for phase

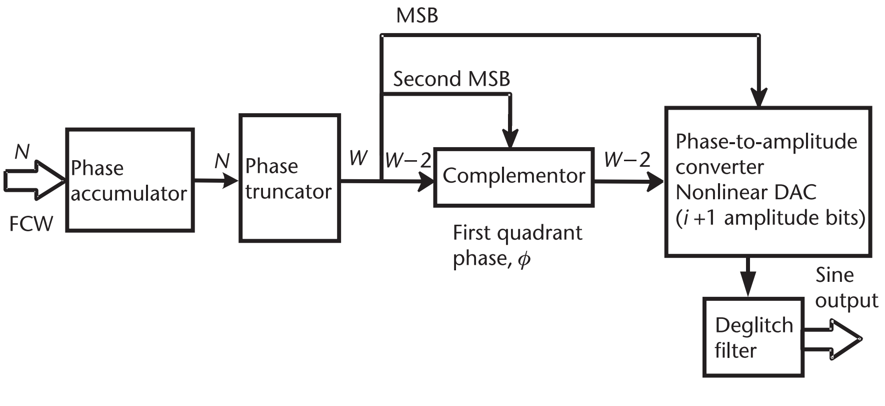


Figure 10.12 High-speed DDS with a nonlinear, sine-weighted DAC.

accumulators with different clock speeds. The pipelined accumulator is used for a constant input word and can achieve the highest operating frequency, while the accumulator with *carry look ahead* (CLA) adders can be employed for variable inputs with medium operating frequencies. Finally, nonlinear DAC designs are discussed.

10.5.1 Pipelined Accumulator

The accumulator delay is dominated by the adder delay. The simplest way to construct an N-bit adder is to place N 1-bit adders in a chain, starting with a 1-bit half adder, followed by $N - 1$ 1-bit full adders, with the carry in of the full adder connected to the carry-out of the previous bit. This ripple adder topology uses the least hardware but operates at the slowest speed. The delay of a ripple adder is due to the propagation of the carry bit from the LSB to the MSB. The sum and carry-out of a full adder can be expressed as

$$\begin{cases} \text{Sum} = A \oplus B \oplus C_{\text{in}} \\ C_{\text{out}} = A \cdot B + B \cdot C_{\text{in}} + C_{\text{in}} \cdot A \end{cases} \tag{10.49}$$

where A and B are the input bits and C_{in} is the carry-in of the adder. The delay of an N-bit ripple adder is hence simply

$$\text{Delay} = (N - 1)T_{\text{carry}} + T_{\text{sum}} \tag{10.50}$$

where T_{carry} is the time for carry generation and is equal to twice the delay of an AND gate. Similarly, T_{sum} is the time for sum generation in a 1-bit adder and is equal to twice the delay of an XOR gate.

If the accumulator input is time invariant, each bit of the input word and the adder output bits can be delayed properly such that an N-bit accumulator can operate at the speed of a 1-bit adder. This type of accumulator, called a *pipelined accumulator*, uses the most hardware but achieves the highest speed. Figure 10.13 illustrates a generic architecture of an $N \times M$ pipelined accumulator with an $(N \times M)$-bit input FCW and a total of M pipelined rows. In the figure, Verilog notation is used to represent the location of the bits. For instance, FCW[N:2N – 1] denotes bits of FCW from the Nth bit to the $(2N - 1)$th bit. Each row has a total of M delay stages placed at the input and output of an N-bit adder. Obviously, an $N \times M$ pipelined accumulator has a latency period equal to the propagation delay of $M - 1$ clock cycles. Note that an accumulator needs at least one delay stage, even without any pipelined stages. The illustrated pipelining accumulator allows the $N \times M$ bit accumulator to operate at the speed of an N-bit accumulator, a speedup of M times. When the number of adder bits is set to one ($N = 1$), the $1 \times M$ bit accumulator can operate at the same speed as a 1-bit adder. As an example, to realize an 8-bit accumulator, we can set $N = 1$ and $M = 8$. Thereby, an 8-bit accumulator runs at the speed of a 1-bit accumulator consisting of a full adder and a flip-flop. Chapter 5 discusses the CML circuit implementations for a 1-bit adder and resetable flip-flops. For example, Figure 5.10 shows a CML adder

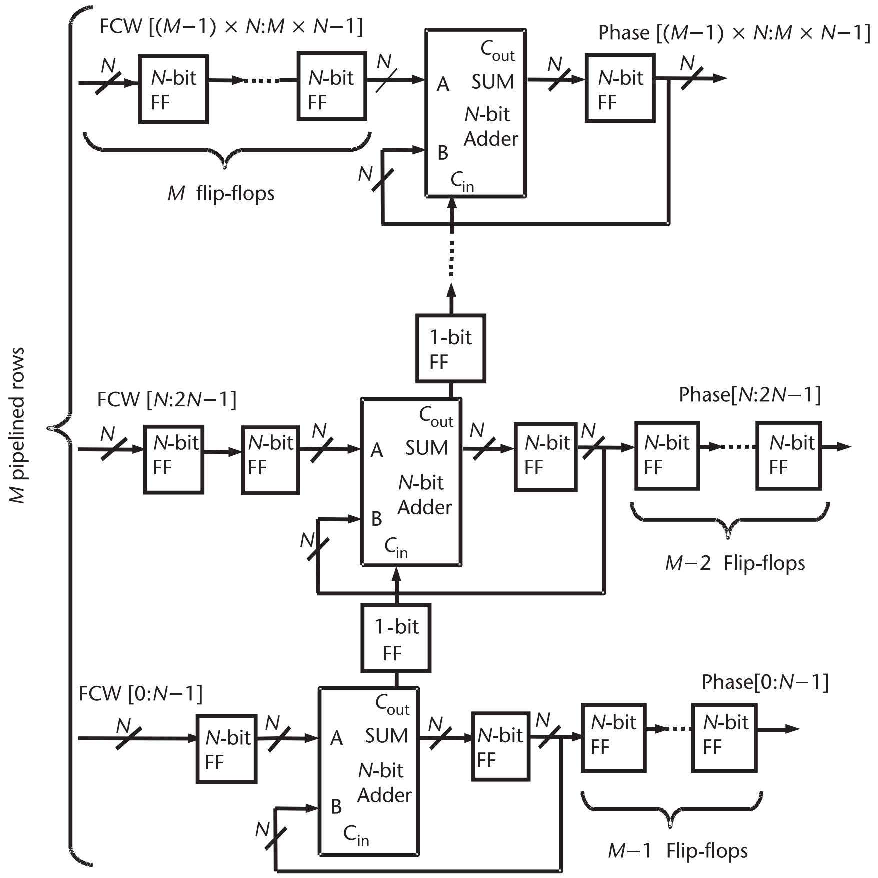


Figure 10.13 Generic architecture of an $N \times M$ pipelined accumulator.

circuit, and Figure 5.17 shows a CML latch with active-low reset using four-level transistors.

10.5.2 Accumulator with CLA Adders

As is discussed in Chapter 11, the DDS architecture can incorporate various types of modulation and waveform generation. To allow the DDS to operate with modulation, the pipelined accumulator needs to be modified to allow variable input FCWs. In some types of modulation, the FCW input of the DDS changes continuously. To incorporate frequency-modulation techniques in DDSs, the latency period (the propagation delay) has to be reduced.

The 8-bit, fully pipelined accumulator architecture has a large latency period of $N - 1 = 7$ clock cycles. This latency can be reduced to one clock cycle using an $N \times M = 4 \times 2$ configuration, as shown in Figure 10.13. The two adders in the accumulator are constructed using a 4-bit CLA architecture whose principles are discussed in the following paragraphs. The latency can be reduced to zero by using an 8-bit CLA adder to configure $N \times M = 8 \times 1$ accumulators. However, reducing the pipelined stages M causes a considerable reduction in speed. The maximum speed attained by the $N \times M = 4 \times 2$ pipelined accumulator with CLA adders is the same as the maximum speed attained by a 4-bit CLA accumulator, which can operate at a maximum speed of about $f_T/12$ using CML logic circuits.

For the adders shown in Figure 5.10, the speed bottleneck of a ripple adder is the carry propagation from the LSB up to the MSB. If the carry propagation delay can be reduced using additional logic, the adder delay can also be reduced. CLA adders divide the full adders into subgroups and employ the CLA logic to speed up the carry propagation process. According to (10.49), the carry out of the ith full adder in an N-bit ripple adder chain can be expressed as

$$c_i = A_i \cdot B_i + A_i \cdot c_{i-1} + B_i \cdot c_{i-1} = A_i \cdot B_i + (A_i \oplus B_i) \cdot c_{i-1} \quad (10.51)$$

where A_i and B_i are the inputs, and c_{i-1} is the carry-in of the ith full adder, respectively. Since the inputs of the adder are available at the beginning of the addition operation, we can extract the following two terms, which can be calculated prior to the arrival of the carry-in bit:

$$\begin{cases} \text{Carry generate: } g_i = A_i \cdot B_i \\ \text{Carry propagate: } p_i = A_i \oplus B_i \end{cases} \quad (10.52)$$

Analyzing (10.51), we conclude that if $g_i = 1$, a carry is generated regardless of the carry-in bit. Similarly, if $p_i = 1$, the carry-in bit is propagated to the carry-out bit. Thus, the two terms g_i and p_i are referred to as the *carry generate* and *carry propagate*, respectively. With the g_i and p_i terms, the sum and carry out of each bit can be found as

$$\begin{cases} \text{Carry-out: } c_i = g_i + p_i \cdot c_{i-1} \\ \text{Sum: } s_i = p_i \oplus c_{i-1} \end{cases} \quad (10.53)$$

Note that the g_i and p_i terms can be calculated in parallel for all bit positions in one gate delay right after the adder inputs are available. The niche of a CLA adder is to use the following additional logic to set the carry bits with only the g_i and p_i information of each bit as

$$\begin{cases} c_0 = g_0 + p_0 \cdot c_{\text{in}} = g_0 \\ c_1 = g_1 + p_1 \cdot g_0 \\ c_2 = g_2 + p_2 \cdot g_1 + p_2 \cdot p_1 \cdot g_0 \end{cases} \quad (10.54)$$

The resulting CLA adder is shown in Figure 10.14, which differs from the simple ripple adder in the sense that the carry bits are generated in the CLA logic unit based on (10.54). Thereby, the full adders (FAs) do not need to wait for their previous bits to generate the carry bits; as a result, the speed is increased. In the design of a high-speed CLA adder using CML logic with a 3.3V power supply, it is typically assumed that the CML logic gates allow a maximum of three inputs, meaning the fan-in of the gate is three. It is also assumed that the XOR gate has

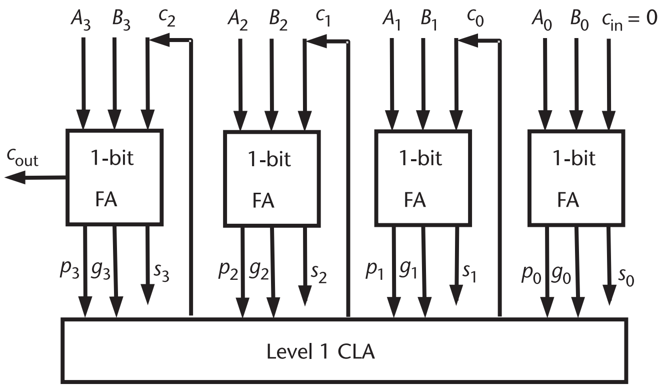


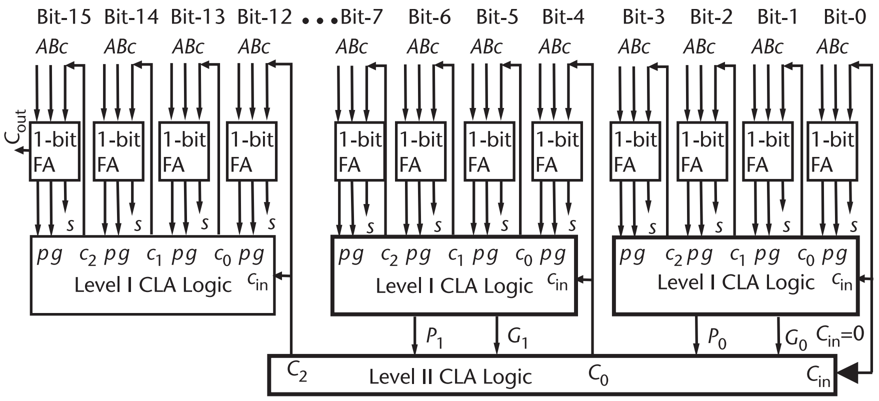
Figure 10.14 A 4-bit, level I, CLA adder configuration.

twice the delay of the AND gate, and the 4-bit CLA adder has a total of six gate delays, allowing all logic expressions in (10.52) to (10.54).

Obviously, CLA logic quickly becomes complicated when the number of input bits increases. When the number of input bits increases to five, the CLA logic requires logic gates with a fan-in of four, which is difficult to implement using high-speed CML circuits. This problem can be solved if we subdivide the input bits into groups and apply additional CLA logic for carry-bit calculation. If we limit the maximum fan-in of the logic gates to four, a 16-bit CLA adder can be constructed as shown in Figure 10.15, where every 4 bits have been grouped as a level I CLA adder, and the four level I CLA adders are further connected to form a 16-bit, level II CLA adder. For the two-level CLA adder, the level I CLA logic can be obtained by

Level I CLA Logic:

$$\begin{cases} c_0 = g_0 + p_0 \cdot c_{in} \\ c_1 = g_1 + p_1 \cdot g_0 + p_1 \cdot p_0 \cdot c_{in} \\ c_2 = g_2 + p_2 \cdot g_1 + p_2 \cdot p_1 \cdot g_0 + p_2 \cdot p_1 \cdot p_0 \cdot c_{in} \end{cases} \tag{10.55}$$



Bit-15 Bit-14 Bit-13 Bit-12 • • • Bit-7 Bit-6 Bit-5 Bit-4 Bit-3 Bit-2 Bit-1 Bit-0
ABc
C_{out}
1-bit FA
s
pg c_2 pg c_1 pg c_0 pg
Level I CLA Logic c_{in}
P_1 G_1 P_0 G_0 C_{in}=0
C_2 Level II CLA Logic C_0 C_{in}

Figure 10.15 A 16-bit, level II, CLA adder configuration.

and the level II CLA logic can be obtained by

Level II CLA Logic:

$$\begin{cases} C_0 = G_0 + P_0 \cdot C_{\text{in}} \\ C_1 = G_1 + P_1 \cdot G_0 + P_1 \cdot P_0 \cdot C_{\text{in}} \\ C_2 = G_2 + P_2 \cdot G_1 + P_2 \cdot P_1 \cdot G_0 + P_2 \cdot P_1 \cdot P_0 \cdot C_{\text{in}} \end{cases} \tag{10.56}$$

The level II carry propagate P_i are obtained by ANDing all of the level I carry propagates p_i inside the group as

$$\begin{cases} P_0 = p_3 \cdot p_2 \cdot p_1 \cdot p_0 \\ P_1 = p_7 \cdot p_6 \cdot p_5 \cdot p_4 \\ P_2 = p_{11} \cdot p_{10} \cdot p_9 \cdot p_8 \end{cases} \tag{10.57}$$

All of the level II carry generates G_i are obtained similarly to the level I carry-out logic:

$$\begin{cases} G_0 = g_3 + p_3 \cdot g_2 + p_3 \cdot p_2 \cdot g_1 + p_3 \cdot p_2 \cdot p_1 \cdot g_0 \\ G_1 = g_7 + p_7 \cdot g_6 + p_7 \cdot p_6 \cdot g_5 + p_7 \cdot p_6 \cdot p_5 \cdot g_4 \\ G_2 = g_{11} + p_{11} \cdot g_{10} + p_{11} \cdot p_{10} \cdot g_9 + p_{11} \cdot p_{10} \cdot p_9 \cdot g_8 \end{cases} \tag{10.58}$$

Note that the level II P_3 and G_3 for the last group are not needed in the 16-bit CLA logic given in (10.56).

Assuming the XOR delay is twice the delay of an AND gate, the above-developed, 16-bit, level II CLA adder with a maximum fan-in of four experiences the following delays: (1) two gate delays to calculate level I carry generate g_i and propagate p_i in (10.52); (2) two gate delays to calculate level II carry generate G_i and propagate P_i in (10.57) and (10.58); (3) two gate delays to calculate level II carry in (10.56); (4) two gate delays to calculate level I carry in (10.55); and (5) two gate delays to calculate the sum and carry-out of the MSB based on (10.53). Hence, the total delay for the 16-bit CLA adder is 10 AND gate delays. In comparison, an N-bit ripple adder experiences $(2N - 1)$ gate delays as follows: (1) one gate delay for the LSB to generate the carry-out; (2) $(N - 2) \times 2$ gate delays for all of the bits except the LSB and MSB to generate the carry-outs; and (3) two gate delays for the MSB to generate the sum and carry-out. Therefore, a 16-bit ripple adder has a total of 31 gate delays, which is 21 more gate delays than the 16-bit CLA adder. When the number of bits is further increased, the CLA adder will have an even larger speed advantage over the simple ripple adder. Recall that a 4-bit, level I CLA has six gate delays. Extending the 4-bit, level I CLA adder to a 16-bit, level II CLA adder requires only four more gate delays to calculate the level II carry generate G_i and propagate P_i in (10.57) and (10.58) and the level II carry-outs in (10.56). Further extending the 16-bit, level II CLA adder to a 32-bit, level III CLA adder requires an additional four gate delays to calculate the level III carry generate G_i and propagate P_i and the level III carry-outs. Thus, a 32-bit, level III

CLA adder has 14 AND gate delays. In comparison, a 32-bit ripple adder has a total of 63 gate delays, which is 49 more gate delays than the 32-bit CLA adder. Note that a 32-bit accumulator is widely used in low- and medium-speed DDS designs.

10.5.3 Sine-Weighted Nonlinear DACs

The typical high-speed DDS utilizes a sine-weighted DAC operating in current mode, which does not require an op-amp buffer at the output; thus, the DDS speed is not limited by the bandwidth of the op-amp. For constructing a high-speed, nonlinear DAC, a current-steering DAC architecture is the ideal candidate since it can generate a Nyquist output signal with high accuracy at a high update rate. The nonlinear DAC contains a current-cell matrix. Each DAC current cell outputs a current proportional to the sine or cosine value of the corresponding phase indicated by the input phase word. The sinusoidal output is obtained by summing the output currents from all the cells through an external pull-up resistor. Dynamic performance of the DDS rapidly degrades with frequency due to transient glitches in the DAC. These glitches can be minimized by using a thermometer decoder, which will ensure that the minimum number of cells switch simultaneously. In addition to the conventional static performance measures, such as offset error, gain error, integral nonlinearity (INL), and differential nonlinearity (DNL), high-speed nonlinear DAC designers are also concerned with maximum operating frequency, power consumption, and dynamic performances, such as spurious-free dynamic range (SFDR) and the signal-to-noise and distortion (SINAD) ratio. SFDR is the difference between the rms power of the fundamental tone and the largest spur within the frequency band of interest. SINAD is the difference between the rms power of the fundamental tone and the noise power and distortion components that fall within the Nyquist frequency band.

Based on the SFDR requirement, the number of input phase bits W and the number of amplitude bits of the nonlinear DAC can be determined. Figure 10.6 gives the difference between the largest spur and the fundamental signal in dBc versus the number of phase bits and the number of DAC amplitude bits. Note that the total output current doubles when the number of DAC bits increases by one bit, and the number of DAC cells also doubles when the number of phase bits is increased by one. Moreover, if the DAC output uses an open-collector resistor, limited voltage headroom may be problematic if the total output current is too large. For low-power applications, it is necessary to choose a reasonable number of DAC output bits based on the required SFDR. Then, the number of phase bits is chosen to be slightly larger than that of the DAC output bits such that the number of DAC bits dominates the overall quantization noise. For example, if the spur magnitude is required to be below –45 dBc, then, according to Figure 10.6, the number of DAC bits needs to be larger than eight. This result is obtained under the assumption that the number of phase bits W is greater than the number of DAC bits. If the number of output DAC bits is known, the number of phase bits W can be obtained by the SINAD requirement. By adding a $\Sigma\Delta$ modulator to the DDS, the quantization noise can be pushed away from the band of interest; thus, the SINAD can be increased.

10.5.4 Nonlinear DAC Segmentations

Although the ROM-less DDS does not have a ROM to store the sinusoidal values, the sine ROM compression algorithm used in the conventional DDS can also be employed to compress the number of current cells in a nonlinear DAC. The ROM-less DDS uses the sinusoidal data to weight the DAC current sources. Hence, the conventional sine compression algorithm can be used to simplify the DAC current units. First, the ROM-less DDS architecture exploits the quadrant symmetry property of the sine function about $\pi/2$ and π. Thus, the two MSBs of the accumulator output denote the different quadrants of the sinusoid as shown in Figure 10.12. The remaining $W - 2$ bits from the complementor represent the first quadrant phase word ϕ. These bits are used to generate the sinusoidal waveform in the first quadrant. The complete sinusoid can be reconstructed by combining the two MSBs and ϕ. To segment the sinusoids further, ϕ can be divided into three parts α (the most significant part), β (the middle part), and γ (the least significant part), where $\phi = \alpha + \beta + \gamma$. The number of bits in segments α, β, and γ are assumed to be a, b, and c, respectively.

Based on trigonometric identities, the first quadrant of the sinusoid can be expressed as [20]

$$\begin{aligned}\sin\frac{\pi(\alpha+\beta+\gamma)}{2(2^{a+b+c}-1)} &= \sin\frac{\pi(\alpha+\beta)}{2(2^{a+b+c}-1)}\cos\frac{\pi\gamma}{2(2^{a+b+c}-1)}\\ &\quad+\cos\frac{\pi(\alpha+\beta)}{2(2^{a+b+c}-1)}\sin\frac{\pi\gamma}{2(2^{a+b+c}-1)}\\ &\approx \sin\frac{\pi(\alpha+\beta)}{2(2^{a+b+c}-1)}+\cos\frac{\pi(\alpha+\beta_{\text{avg}})}{2(2^{a+b+c}-1)}\sin\frac{\pi\gamma}{2(2^{a+b+c}-1)}\end{aligned} \tag{10.59}$$

Equation (10.59) indicates that the sine function can be implemented using a coarse DAC representing the first term in (10.59), which contains only α and β, and a fine DAC representing the second term in (10.59), which is dependent not only on α and β, but also on γ. The first term is monotonic and can be realized as a coarse, nonlinear DAC using the full thermometer code, and the second term can be realized using a fine nonlinear DAC. If the nonlinear DAC has $i + 1$ bits of amplitude resolution, each coarse DAC current cell output (O_k) can be expressed as [21]

$$O_k = \begin{cases} \text{int}\left[(2^i - 1)\sin\dfrac{\pi}{2(2^{a+b}-1)}\right] & \text{for } k = 0 \\ \text{int}\left[(2^i - 1)\sin\dfrac{\pi(2k+1)}{2(2^{a+b}-1)} - \sum\limits_{n=0}^{k-1} O_n\right] & \text{for } 1 \le k \le 2^{a+b} - 1 \end{cases} \tag{10.60}$$

where int[ ] denotes rounding to the nearest integer, and the index k is given by

$$k = \frac{\alpha + \beta}{2^c} \tag{10.61}$$

As the number of phase bits increases, the number of DAC current cells increases; therefore, the power consumption and die area increase as well. Since the number of bits in segments α and β is less than the total number of input phase bits without segmentation, the number of coarse DAC cells is much less than that required without segmentation. The second term in (10.59) forms the fine nonlinear DAC to interpolate the amplitudes between adjacent coarse DAC outputs. The value of β in (10.59) has been approximated in (10.61) using the average value, β_{avg}, such that the fine DAC current depends only on α and γ, which reduces the number of fine DAC cells. The fine DAC output is not monotonic in α and γ. Thus, a fine DAC is constructed using $2^a - 1$ nonlinear sub-DACs. The mth DAC cell output, $O_{\alpha,m}$ in the αth sub-DAC can be approximated as

$$O_{\alpha,m} \approx \begin{cases} \operatorname{int}\left[(2^i - 1)\cos\dfrac{\pi(\alpha + \beta_{\text{avg}})}{2(2^{a+b+c} - 1)}\sin\dfrac{\pi}{4(2^{a+b+c} - 1)}\right] & \text{for } m = 0 \\ \operatorname{int}\left[(2^i - 1)\cos\dfrac{\pi(\alpha + \beta_{\text{avg}})}{2(2^{a+b+c} - 1)}\sin\dfrac{(2m+1)\pi}{4(2^{a+b+c} - 1)} - \sum_{n=0}^{m-1} O_{\alpha,n}\right] & \text{for } 1 \le m \le 2^c - 1 \end{cases} \tag{10.62}$$

The total DAC output is the sum of the current outputs of the coarse DAC and the fine sub-DACs. Figure 10.16 illustrates the resultant DDS with segmented nonlinear DACs. Different segmentation gives different performance due to the amplitude errors in the approximation. For a nonlinear DAC with 12-bit phase

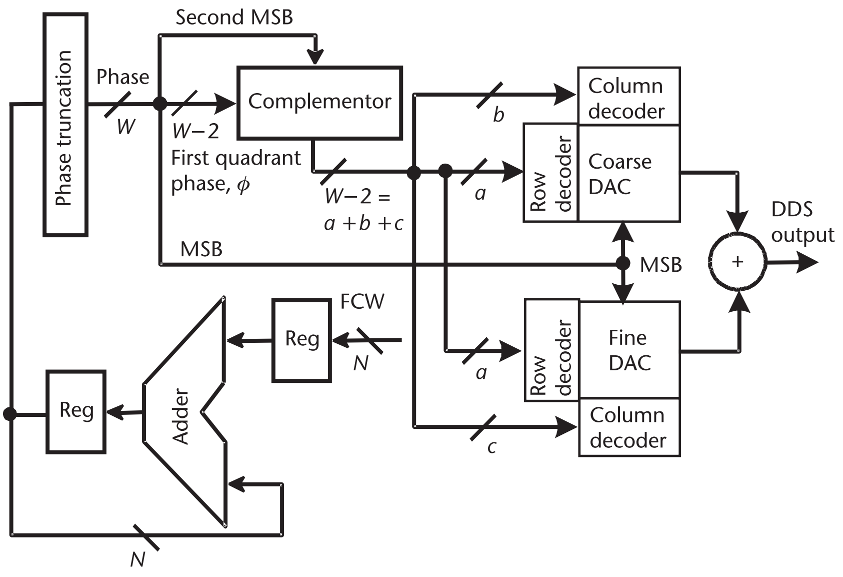


Figure 10.16 A DDS architecture with segmented nonlinear DACs.

input and 11-bit amplitude resolution, a segmentation of $a - b - c = 3 - 4 - 3$ gives the optimal results with minimum error, small die size, and minimum power consumption [20].

10.5.5 Nonlinear Coarse DAC

If the number of phase bits is not large (for example, smaller than 8 bits), the segmentation of the sine-weighted DAC is not necessary. The nonlinear DAC can be constructed using just a coarse DAC. It is assumed that the nonlinear DAC output has $i + 1$ bit resolution and that its input phase word has W bits. The W-bit phase word from the phase accumulator is split into the 2 MSB bits and the rest of $(W - 2)$ LSB bits. The $(W - 2)$ LSBs are fed into a complementor, which is a one's complementor; that is, it flips each bit of the $(W - 2)$-bit phase word. The complementor output is separated into a MSBs and b LSBs, which are used to control the row and column thermometer decoder. These decoders map the binary inputs into thermometer code outputs that are used to switch the DAC current cells. Each DAC cell output current (O_k) can be obtained from (10.60). In the partition, the first quadrant of the sinusoid is divided into 2^{a+b} phase steps with k representing the phase step index between 0 and $\pi/2$. The DAC output at phase step k is the sum of the outputs of the DAC current cell from 0 to k according to (10.60). The architecture to realize the coarse nonlinear DAC with $2^{a+b} - 1$ current cells is illustrated in Figure 10.17.

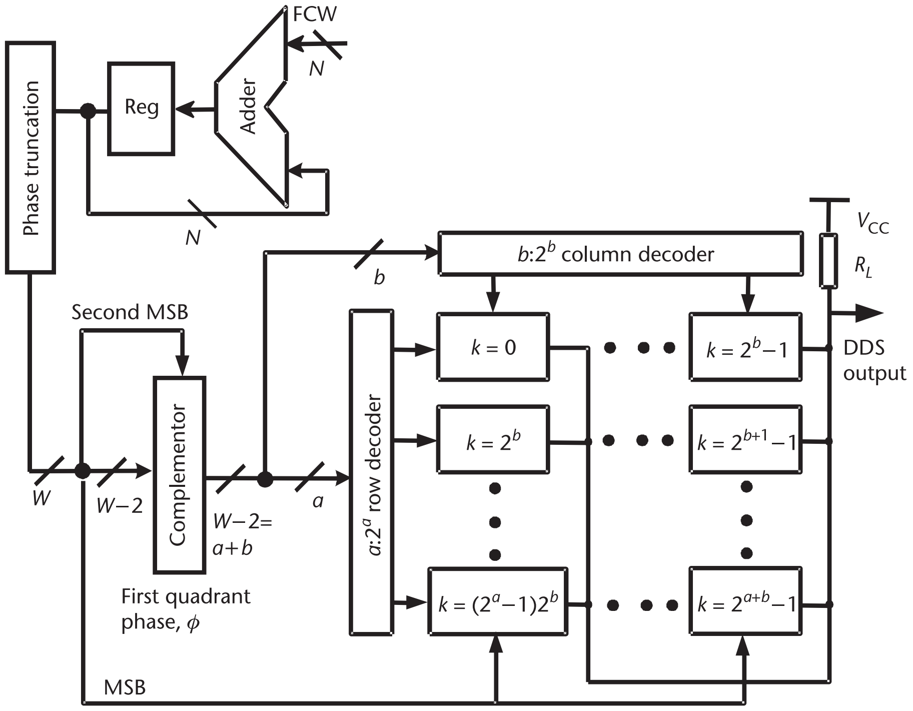


Figure 10.17 A DDS architecture with a nonlinear, coarse DAC.

Figure 10.18 shows the DAC cell with different current sources O_k, where n represents the number of duplicated unit current sources that form the desired current value O_k given in (10.60). A CML topology is used to implement the high-speed current switches. Each current source is biased close to the peak-f_T current in order to achieve high-speed operation, and minimum size transistors are used for low power consumption. The bias current should be carefully chosen, considering the speed, power consumption, and DAC output full-scale voltage swings. High bias current is not preferable because the total power consumption increases, and the output pull-up resistor value decreases in order to keep the same full-scale output voltage. The accuracy of the bias current is ensured using optimized bandgap references driving cascode current sources. Within a sampling period, the current is held constant. The two pairs of current switches, A and B, are used for producing the positive and negative regions of the sine outputs, respectively. For the positive region, the signal MSB is low, and the current switch pair B in all the DAC cells is turned on. The thermometer code decoders will turn on the pair A according to cell control logic. For the negative region, the signal MSB is high, and the current switch pair A in all the DAC cells is turned off. The thermometer code decoder will turn on the cell's switch pair B according to the logic control circuit shown in Figure 10.18. In addition, all switching control signals are buffered to ensure differential synchronous switching for all cells. Notice that the current switches are built using bipolar transistors to achieve the high-speed switching.

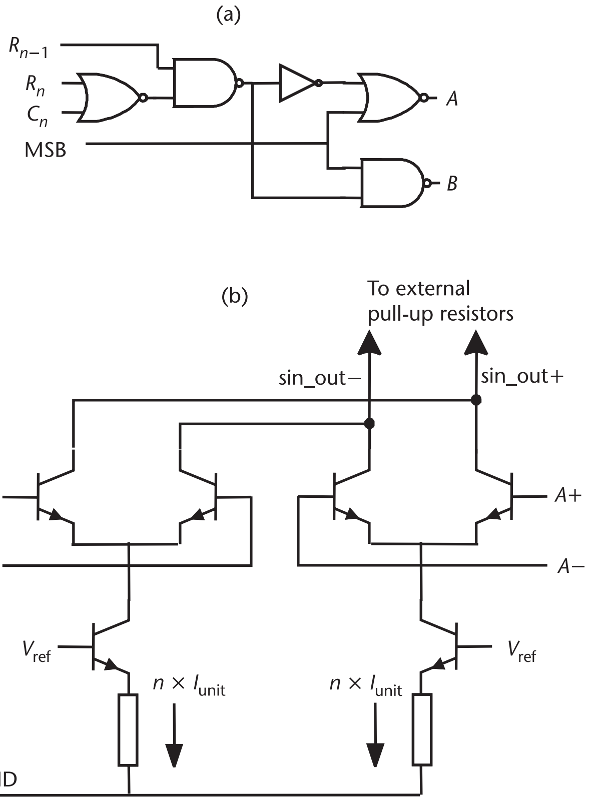


Figure 10.18 DAC current cell circuitry: (a) control logic, and (b) current switch.

In a thermometer-coded DAC architecture, every current source has a weight of one LSB. The current source switches are controlled by the thermometer decoder output. Table 10.1 gives the thermometer decoder logic truth table. As shown, the thermometer code leads to only one additional current source transition when the LSB of the input binary phase word changes. Thus, the DAC has a guaranteed monotonic behavior, which results in good DNL error and small dynamic switching errors. The major disadvantage of the thermometer decoded DAC is its complicated thermometer decoder with large size and power consumption, especially when the number of bits is higher than 10. In general, a P-bit binary input phase word will be mapped to 2^P-bit thermometer decoder output. In the nonlinear DAC architecture, the sinusoidal output is obtained by summing the output currents from all the selected current cells. The thermometer decoding reduces dynamic errors by ensuring that the minimum number of cells switch simultaneously. The number of current sources that are turned on should be equal to the value of the thermometer input code.

Example 10.2: The Design of Current Sources for a Sine-Weighted Nonlinear DAC
Design the current sources for a sine-weighted nonlinear DAC with 8-bit phase and 8-bit amplitude resolution.

Solution: For 8-bit phase resolution, $W = 8$. The two MSBs are used to select the quadrants of the sinusoid, and the remaining $W - 2 = 6$ bits are used to control the DAC current cells, as shown in Figure 10.17. Thus, $a = b = 3$, and there is a total of $2^3 \times 2^3 = 8 \times 8 = 64$ current cells. According to (10.60), the sine-weighted current O_k in each current cell is given by

$$O_k = \begin{pmatrix} 2 & 4 & 6 & 6 & 6 & 5 & 3 & 0 \\ 0 & 3 & 5 & 6 & 6 & 6 & 4 & 2 \\ 2 & 4 & 6 & 6 & 6 & 5 & 2 & 1 \\ 1 & 3 & 5 & 6 & 6 & 6 & 3 & 2 \\ 3 & 4 & 6 & 3 & 6 & 5 & 3 & 2 \\ 1 & 3 & 5 & 6 & 7 & 5 & 4 & 1 \\ 2 & 4 & 5 & 6 & 6 & 5 & 3 & 1 \\ 0 & 4 & 4 & 6 & 7 & 5 & 5 & 1 \end{pmatrix} \tag{10.63}$$

Table 10.1 Thermometer Code Representation for 3-Bit Binary Phase Word

Decimal Phase (k)	*Binary Phase A3 A2 A0*	*Thermometer Decoder Output R6 R5 R4 R3 R2 R1 R0*
0	0 0 0	0 0 0 0 0 0 0
1	0 0 1	0 0 0 0 0 0 1
2	0 1 0	0 0 0 0 0 1 1
3	0 1 1	0 0 0 0 1 1 1
4	1 0 0	0 0 0 1 1 1 1
5	1 0 1	0 0 1 1 1 1 1
6	1 1 0	0 1 1 1 1 1 1
7	1 1 1	1 1 1 1 1 1 1

Note that the above weighted current represents the number of unit current sources in each current cell. In the layout, all of the current cells should have the same height. The sum of each row represents the total number of unit current cells in the row. Thus, the sum of each row in (10.63) is designed to be the same in order to achieve the same layout height and length for each row.

The layout compactness of the current sources is important since it reduces the integral nonlinearity of the nonlinear DAC due to symmetry and minimizes gradient errors. To compensate for symmetry and gradient errors caused by variations in the temperature and process mismatch, a special switching scheme based on double centroid and random walking [22, 23] can be implemented. There are 64×2 sine-weighted current sources to be placed as close as possible in layout to achieve the best possible matching. The current matrix includes 256×2 unit current sources, and each provides the minimum current required to achieve the specified speed.

10.5.6 Comparison of ROM-Less DDS Performance

Ultra-high-speed, ROM-less DDS can be implemented in InP, SiGe, and CMOS technologies. To choose a proper technology for high-speed DDS implementation, let us briefly review the pros and cons of the three technologies. Lateral and vertical device scaling are common ways to increase a heterojunction bipolar transistor's (HBT) high-frequency performance. These techniques are often limited by molecular beam epitaxy and processing technology. An InP HBT achieves high cutoff frequency f_T by scaling the device vertically with a trade-off in base-collector capacitance (C_{BC}) and ultimately the device $f_{\max}$. State-of-the-art lithography technology can be used laterally to scale the device, but, eventually, the high-frequency performance will be limited by extrinsic parasitic parameters. State-of-the-art InP technology can achieve an f_T of 137 GHz and an $f_{\max}$ of 267 GHz [16]. However, InP devices consume much more power, as Table 10.2 indicates. A SiGe HBT is similar to a silicon bipolar transistor except for the base. Doping Ge into the base of a silicon bipolar transistor increases the electron mobility and, thus, increases the

Table 10.2 Comparison of DDS Designs Using InP, SiGe, and CMOS Technologies

Technology	*InP*	*0.5-μm SiGe*	*0.18-μm SiGe*	*0.35-μm CMOS*
$f_T/f_{\max}$ (GHz)	*137/120* [16]	*47/65* [17]	*120/100*	*27/35* [18]
Emitter area of minimal size transistor (μm^2)	1.5×4	0.5×1	0.2×0.64	0.35×0.4
Emitter current density at peak f_T (mA/μm^2)	1 ~ 1.2	1.2	6	1.8
Peak f_T current of minimum-size transistor (mA)	7.2	0.6	0.77	0.28
B_{vceo}	8V	3.3V/5.5V	1.8V/4.25V	
Accumulator size	8 bit	8 bit	8 bit	8 bit
Nonlinear DAC bits	8 bit	8 bit	8 bit	8 bit
Maximum clock frequency	9.2 GHz	5 GHz	10 GHz	2 GHz
Power consumption	15W	3W	2.4W	0.82W
Die size (mm^2)	8×5	2×3	1.4×3	1.9×2.1
Power efficiency (GHz/W)	0.61	1.67	4.2	2.4
Area efficiency (GHz/mm^2)	0.23	0.83	2.38	0.5

device f_T. Meanwhile, the base resistance is also reduced, leading to lower thermal noise. In addition, the Ge composition is typically graded across the base to create an electric field for accelerating minority carriers moving across the base. The transistor gain of a SiGe HBT is also increased compared to a silicon BJT, which can then be traded for better linearity and lower power consumption. Compared to an identically constructed Si BJT under the same bias current, a SiGe HBT has higher f_T, higher gain, higher output impedance, lower RF noise, and lower $1/f$ noise. CMOS devices have advantages such as superior linearity and low-voltage operation due to lower threshold voltages, while SiGe HBT devices offer the advantages of excellent speed, good noise performance, and improved transconductance. The $1/f$ noise due to random trapping of charges at the oxide-silicon interfaces and thermal noise due to gate and channel resistances are both significantly higher in CMOS than in SiGe HBTs. To reduce noise, large CMOS devices and large operating currents are often required, which leads to reduced speed and increased power consumption.

A ROM-less DDS was implemented in a 0.35-μm CMOS technology [18], which consumes a total of 820 mW at 2 GHz with 3.3V power supply. A DDS implemented in a 137-GHz InP technology operating at a world-record clock frequency of 9.2 GHz was reported in [16]. However, the InP DDS consumes 15W of power and requires a die size of 40 mm^2. As Table 10.2 shows, the minimum transistor size in the InP technology is much larger than standard CMOS or SiGe technologies. Although the current density to achieve the peak f_T frequency in InP, SiGe, and CMOS are similar, the current needed to operate the minimum transistor close to peak f_T frequency is a lot lower in SiGe and CMOS technologies. Compared to the SiGe technologies, the InP technology requires 10 times more current to operate the minimum-size transistor at the peak f_T. A 0.18-μm SiGe DDS monolithic microwave integrated circuit (MMIC) with equivalent performance has a power consumption of 2.4W, which is less than one-sixth the power consumption of the InP DDS MMIC. Table 10.2 summarizes the performance comparison between the DDS implementations using InP, SiGe, and CMOS technologies.

References

[1] Vankka, J., "Spur Reduction Techniques in Sine Output Direct Digital Synthesis," *Proc. 50th Annual Frequency Control Symp.*, Orlando, FL, September 1996, pp. 951–959.

[2] Flanagan, M. J., and G. A. Zimmerman, "Spur-Reduced Digital Sinusoid Synthesis," *IEEE Trans. Comm.*, Vol. 43, No. 7, July 1995, pp. 2254–2262.

[3] Flanagan, M. J., and G. A. Zimmerman, "Spur-Reduced Digital Sinusoid Generation Using Higher-Order Phase Dithering," *27th Annual Asilomar Conf. on Signals, Systems, and Computers*, November 1993, pp. 826–830.

[4] Reinhardt, V. R., "Spur Reduction Techniques in Direct Digital Synthesizers," *Proc. 47th Annual Frequency Control Symp.*, Salt Lake City, UT, June 1993, pp. 230–241.

[5] Jenq, Y. C., "Direct Digital Synthesizer with Jittered Clock," *IEEE Transaction on Instrument and Measurement*, Vol. 40, No. 3, June 1997, pp. 653–655.

[6] Song, Y., and B. Kim, "A 14-b Direct Digital Frequency Synthesizer with Sigma-Delta Noise Shaping," *IEEE J. Solid-State Circuits*, Vol. 39, May 2004, pp. 847–851.

[7] Nicholas, H. T., and H. Samueli, "An Analysis of the Output Spectrum of Direct Digital Frequency Synthesizers in the Presence of Phase-Accumulator Truncation," *Proc. 41st Annual Frequency Control Symp.*, Philadelphia, PA, May 1987, pp. 495–502.

[8] Kim, B., H. T. Nicholas, and H. Samueli, "The Optimization of Direct Digital Frequency Synthesizer in the Presence of Finite Word Length Effects," *Proc. 42nd Annual Frequency Control Symp.*, Baltimore, MD, June 1988, pp. 357–363.

[9] Nicholas, H. T., "The Determination of the Output Spectrum of Direct Digital Frequency Synthesizers in the Presence of Phase Accumulator Truncation," Master's Thesis, University of California, Los Angeles, 1985.

[10] Jenq, Y. C., "Digital Spectra of Nonuniformly Sampled Signals: Fundamentals and High-Speed Waveform Digitizers," *IEEE Transaction on Instrument and Measurement*, Vol. 37, No. 6, June 1988, pp. 245–251.

[11] Vankka, J., and K. Halonen, *Direct Digital Synthesizers*, Dordrecht, Netherlands: Kluwer Academic Publishers, 2002.

[12] Jenq, Y. C., "Digital Spectra of Nonuniformly Sampled Signals: Digital Look-Up Tunable Sinusoidal Oscillators," *IEEE Trans. on Instrument and Measurement*, Vol. 37, No. 6, September 1988, pp. 358–362.

[13] O'Leary, P., and F. Maloberti, "A Direct-Digital Synthesizer with Improved Spectral Performance," *IEEE Trans. Comm.*, Vol. 39, No. 7, July 1991, pp. 1046–1048.

[14] Ni, W., et al., "A Direct Digital Frequency Synthesizer with Single-Stage $\Delta\Sigma$ Interpolator and Current-Steering DAC," *IEEE Symposium on VLSI Circuits*, Kyoto, Japan, June 2005, pp. 56–59.

[15] Mortezapour, S., and E. K. F. Lee, "Design of Low-Power Frequency Synthesizer Using Nonlinear Digital-to-Analog Converter," *IEEE J. Solid-State Circuits*, Vol. 34, October 1999, pp. 1350–1359.

[16] Gutierrez-Aitken, A., et al., "Ultrahigh-Speed Direct Digital Synthesizer Using InP DHBT Technology," *IEEE J. of Solid State Circuits*, Vol. 37, September 2002, pp. 1115–1121.

[17] Dai, F. F., et al., "A Low Power 5 GHz Direct Digital Synthesizer Implemented in SiGe Technology," *IEEE 5th Topical Meeting on Silicon Monolithic Integrated Circuits in RF Systems*, Atlanta, GA, September 2004, pp. 21–24.

[18] Yu, X., et al., "2 GHz 8-Bit CMOS ROM-Less Direct Digital Frequency Synthesizer," *IEEE International Symposium on Circuits and Systems (ISCAS)*, Kobe, Japan, May 2005, pp. 4397–4400.

[19] Tierney, J., C. M. Rader, and B. Gold, "A Digital Frequency Synthesizer," *IEEE Trans. on Audio Electroacoust.*, Vol. AU-19, 1971, pp. 48–57.

[20] Jiang, J., and E. Lee, "A Low-Power Segmented Nonlinear DAC-Based Direct Digital Frequency Synthesizer," *IEEE J. of Solid-State Circuits*, Vol. 37, October 2002, pp. 1326–1330.

[21] Mortezapour, S., and E. K. F. Lee, "Design of Low-Power ROM-Less Direct Digital Frequency Synthesizer Using Nonlinear Digital-to-Analog Converter," *IEEE J. Solid-State Circuits*, Vol. 34, October 1999, pp. 1350–1359.

[22] Geert, A., et al., "A 14-Bit Intrinsic Accuracy Q^2 Random Walk CMOS DAC," *IEEE J. Solid-State Circuits*, Vol. 34, December 1999, pp. 1708–1718.

[23] Van den Bosch, A., et al., "A 10-Bit 1-GSample/s Nyquist Current-Steering CMOS D/A Converter," *IEEE J. Solid-State Circuits*, Vol. 36, March 2001, pp. 315–324.

CHAPTER 11

Direct Modulation in Frequency Synthesizers

11.1 Introduction

This chapter introduces techniques to produce digital modulation using synthesizers. There are many types of modulation (ways of encoding data onto a carrier for transmission) in use today. In some applications, the data rate is low; therefore, a simple modulation scheme is adequate and preferred as it reduces the requirements on the radio. Modulation is required in transmission because it is not feasible to build an antenna that will transmit baseband. Instead, the information is first *modulated* around a carrier. Usually the bandwidth that the data occupies is a small fraction of the frequency at which it is transmitted.

In digital communications, a transmitted sequence of bits represents the information. In the simplest case, at low data rates, one bit at a time may be transmitted. In general, the data stream is random; therefore, it has power over a range of frequencies. By comparison, a pure square wave has power at one frequency and at odd harmonics of that frequency. The PSD of an NRZ data stream of bit rate T_B at baseband can be approximated as

$$\mathrm{PSD}_{\mathrm{BB}}(f) = A \cdot T_B^2 \frac{\sin^2(\pi f T_B)}{(\pi f T_B)^2} \tag{11.1}$$

where A is a constant of proportionality. Equation (11.1) is plotted in Figure 11.1. Note that most of the power in the signal is concentrated in frequencies below the bit frequency $f_B = 1/T_B$. Thus, it is often acceptable to limit the spectrum of the transmitted information to 1.4 times f_B and still have the system function properly. If we assume that the signal is limited to a frequency of f_B, then the transmission efficiency is 1 bit/sec/Hz. In order to transmit more data in a given bandwidth, instead of transmitting bits, we can transmit symbols that represent 2 bits or more. In such a case, (11.1) and Figure 11.1 should be thought of as showing the symbol bandwidth and symbol frequency rather than the bit bandwidth and bit frequency. Thus, if symbols are used that represent 2 bits, the transmission efficiency will be roughly 2 bits/sec/Hz, and if a symbol represents 4 bits, then the transmission efficiency will be 4 bits/sec/Hz.

Now, let us suppose that a radio has a bandwidth of BW and a total rms noise power of N. The PSD of the noise is given by

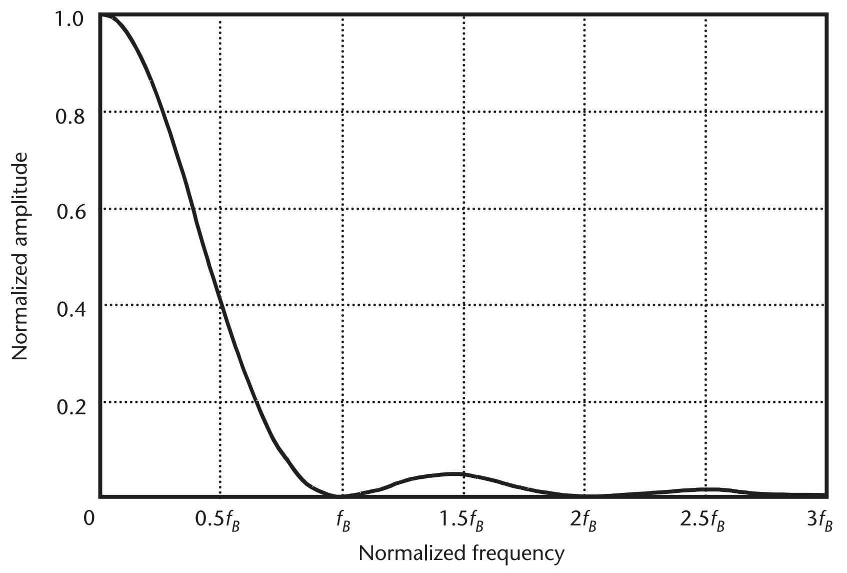


Figure 11.1 Power spectral density of a data stream at base band.

$$N_0 = \frac{N}{\mathrm{BW}} \tag{11.2}$$

Now the energy per bit is the average signal power S multiplied by the time period T_B over which the bit is transmitted:

$$E_b = S \cdot T_B \tag{11.3}$$

Thus, the ratio of E_b to the PSD of the noise N_o is given by

$$\frac{E_b}{N_0} = \frac{S \cdot T_B}{\frac{N}{\mathrm{BW}}} = \frac{S}{N} \cdot \frac{\mathrm{BW}}{f_B} \tag{11.4}$$

Thus, the ratio of the energy per bit to the noise density is equal to the signal-to-noise ratio (SNR) of the radio if the radio bandwidth is equal to the bit frequency. This result is important because it will allow us to relate concepts in digital modulation to the SNR requirement of the radio.

As we have discussed data and the concept of symbols, the next sections describe how various types of synthesizers can be used to produce these waveforms directly.

11.2 Direct Modulation in PLL Frequency Synthesizers

Frequency synthesizers are used to generate carrier frequencies in communication systems. The information to be transmitted or received is modulated onto the carrier frequency by various modulation schemes. As briefly discussed in Chapter 1, modulation of the baseband signal onto the RF carrier can be implemented either in the analog domain using analog mixers or in the digital domain using digital multipliers. As shown in Figures 1.1, 1.3, and 1.8, quadrature modulators or demodulators require quadrature LO signals. While a DDS can directly generate quadrature outputs [1, 2], a PLL synthesizer can generate quadrature LOs by means of the following techniques: (1) a divide-by-two frequency divider following a VCO running at double the frequency, (2) a VCO followed by a passive polyphase RC filter, and (3) a quadrature VCO as discussed in Chapter 8.

In addition to the conventional modulation schemes, the baseband data can also be directly modulated onto the carrier inside frequency synthesizers. Direct modulation can be implemented in PLL synthesizers [3, 4] or, more conveniently, in DDSs [1, 5, 6]. Figure 11.2 shows a direct-modulation scheme through a PLL frequency synthesizer. The binary data stream is first filtered using a digital finite impulse response (FIR) filter. The FIR filter can be programmed for various modulations by programming the filter coefficients. For quadrature phase shift keying (QPSK) modulation, a Nyquist FIR filter can be used for data shaping. For Gaussian frequency shift keying (GFSK) or Gaussian minimum shift keying (GMSK), a Gaussian filter can be implemented for pulse shaping. Note that Section 11.3 describes various types of modulations in more detail. The filtered baseband signal in frequency format directly modulates the carrier fine-tune frequency word in a PLL fractional-*N* synthesizer. The coarse-tune and fine-tune frequency words control the loop divisor through a multimodulus divider (MMD), which forces the loop to lock to the desired output frequency. The synthesizer controls the VCO

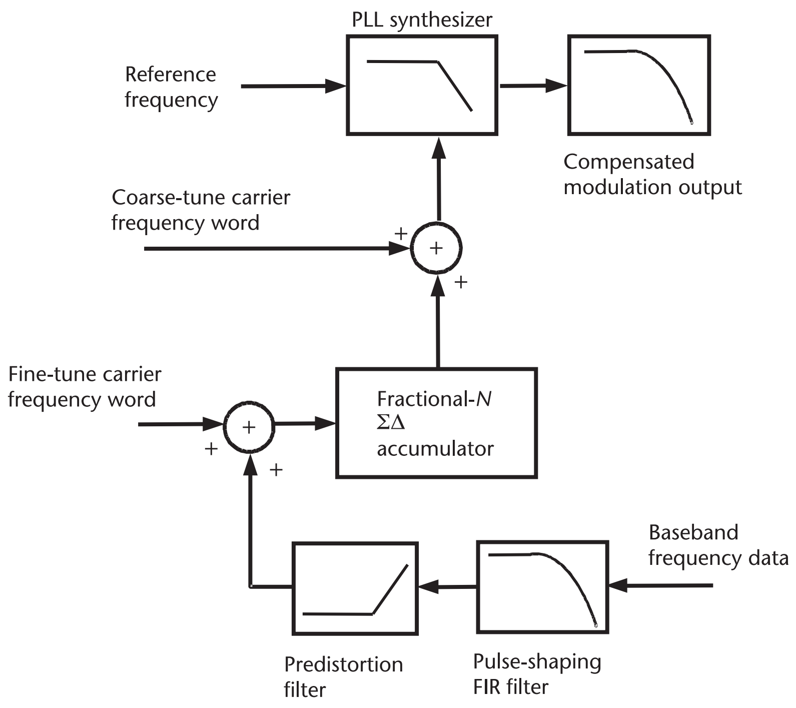

PLL synthesizer
Reference frequency
Compensated modulation output
Coarse-tune carrier frequency word
+
+
+
Fine-tune carrier frequency word
+
+
+
Fractional-*N* ΣΔ accumulator
Baseband frequency data
Predistortion filter
Pulse-shaping FIR filter

Figure 11.2 Direct modulation through a PLL frequency synthesizer.

continuously during the modulation and, thus, eliminates the frequency drift due to the modulation.

The direct-modulation scheme in a PLL simplifies the transmitter architecture by eliminating the need for digital-to-analog converters (DACs), image rejection mixers, RF filters, and other expensive analog components. However, its drawback is its limited modulation bandwidth. As shown in Chapter 3, the PLL bandwidth is of the order of 10 kHz to 1 MHz for a typical PLL synthesizer. Since the baseband data can be modulated only within the lowpass characteristics of a PLL, the maximum modulation bandwidth (i.e., the baseband data rate) is limited by the PLL bandwidth. Recent research has proposed the use of predistortion schemes, which apply an inverse transfer function of the PLL to the baseband signal before it modulates the carrier frequency word [4], as illustrated in Figure 11.2. The direct modulation would be less limited by the narrow PLL bandwidth; however, the predistortion filter and the PLL loop would have to be well matched to avoid distortion in the modulated signal. The matching normally requires automatic calibration so that the PLL can compensate for process and temperature variations [7, 8]. In order to achieve sufficient resolution for the modulation, more bits for the baseband frequency data are required. Direct modulation normally requires a large fractional-N accumulator in order to handle a large number of bits for the modulated frequency word. A large accumulator causes the fractional spurs to be very close to the carrier; hence, high-order $\Sigma\Delta$ noise shaping is required for spurious component reduction. As shown in Chapter 9, the order of the PLL loop filter should be larger than the order of the $\Sigma\Delta$ noise shaping for the loop to filter out the spurious components at higher-frequency offsets. Therefore, a PLL frequency synthesizer used for direct modulation may end up with a high-order loop and a high-order $\Sigma\Delta$ noise shaper, which complicates the design. The spurs, phase noise, and loop dynamics associated with direct modulation in a $\Sigma\Delta$ fractional-N synthesizer can be analytically modeled [9].

Figure 11.3 further illustrates an implementation of a linearized direct modulation through a $\Sigma\Delta$ fractional-N frequency synthesizer [10]. The architecture comprises a baseband modulator, a direct frequency modulator, and a $\Sigma\Delta$ fractional-N PLL synthesizer. The baseband modulator starts with the data to be transmitted (TX data). The demultiplexer (Demux) splits the TX data into in-phase (I) data and quadrature phase (Q) data. The I and Q data are coupled to the respective FIR filters for pulse shaping. The filtered I and Q data are then fed into a lookup table, which stores the $\arctan(Q/I)$ function. Hence, the lookup table generates the modulated phase data, which is further converted to a modulated frequency word by using a digital differentiator d/dt. The output of the differentiator is then coupled into a FIR filter/scaler that functions to predistort the frequency word, compensating for the narrow lowpass dynamics of the PLL and scaling the frequency word such that it is compatible with the frequency scale of the fine-tune frequency word of the synthesizer. Next, the fine-tune frequency word and the modulated and pulse-shaped baseband data stream are mixed together by an adder to provide the frequency-control input for the fractional-N synthesizer.

The modulated frequency word goes through a $\Sigma\Delta$ fractional-N accumulator, and its output further modulates the carrier frequency specified by the coarse-tune word. As discussed in previous chapters, the fine-tune frequency word K controls

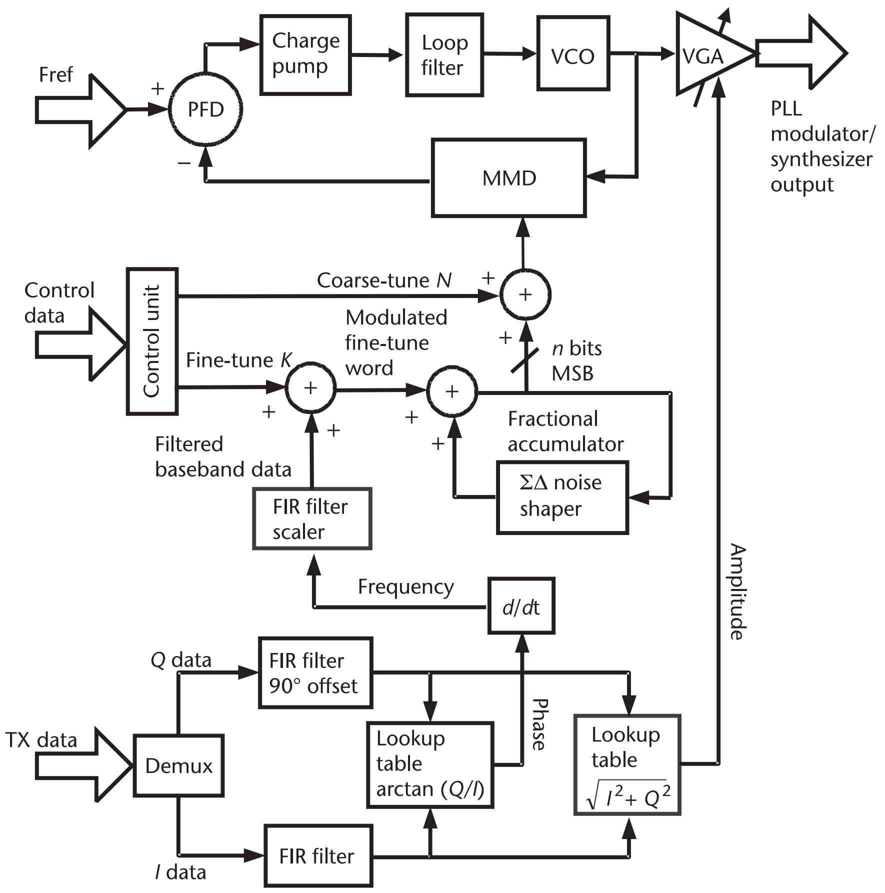


Figure 11.3 A linearized direct modulator utilizing a ΣΔ fractional-*N* frequency synthesizer.

the fractional part of the loop divisor, while the coarse-tune word N controls the integer part of the loop divisor. Now the data is finally modulated onto the desired carrier frequency word, which further controls the loop divisor through an MMD. Thus, ultimately, the PLL forces the VCO output to be continuously adjusted such that the synthesizer output signal is frequency-modulated in accordance with the output signal of the baseband modulator.

In the above configuration, the modulated amplitude information can also be extracted using a second ROM that stores the function of $\sqrt{I^2 + Q^2}$. This amplitude information can be utilized to modulate the output amplitude of the synthesizer through a variable gain amplifier (VGA). The amplitude information can also be included at the last stage prior to the transmission [e.g., a power amplifier (PA) in a transmitter]. Linearized modulation, even with all the modulated components, can be provided by a PA in the saturated mode, which saves power and hardware.

11.3 Direct Digital Modulation and Waveform Generation in a DDS

In addition to frequency synthesis, a DDS can implement various types of modulation and waveform generation since its circuitry can be used to program the frequency, phase, and amplitude of a waveform, as shown in Figure 11.4. With the availability of single-chip high-speed DDSs, such modulation can be done with high accuracy and at high speed, potentially eliminating costly analog modulators associated with communications receivers. For example, radar operational requirements include ever-increasing demands for the generation of affordable, low-noise signals and highly complex, wide-bandwidth waveform generation. These waveforms are used for high-range resolution radars in sorting targets from clutter and low-probability-of-intercept communication applications. A DDS can allow for novel transmitter architectures, for example, generating such waveforms at the highest possible frequency instead of down near baseband. This has the potential advantage of considerably reducing the transmitter size, weight, power requirements, and cost. Note that waveform generation is a unique feature of the DDS approach.

While a PLL synthesizer can synthesize an RF frequency with low phase noise, a DDS can easily generate modulated waveforms with fine resolution. A DDS can be combined into a multiloop hybrid PLL arrangement where both high-resolution and high-frequency RF waveforms can be generated. To overcome the drawback of low output frequency, a PLL can be used to generate a carrier with coarse tuning, and a DDS can be used to fine-tune the carrier frequency, as well as to perform various types of modulation. To accomplish this, the modulated DDS output is mixed with the PLL carrier frequency using quadrature mixers with image rejection. The DDS gives the combination synthesizer small step size, good phase

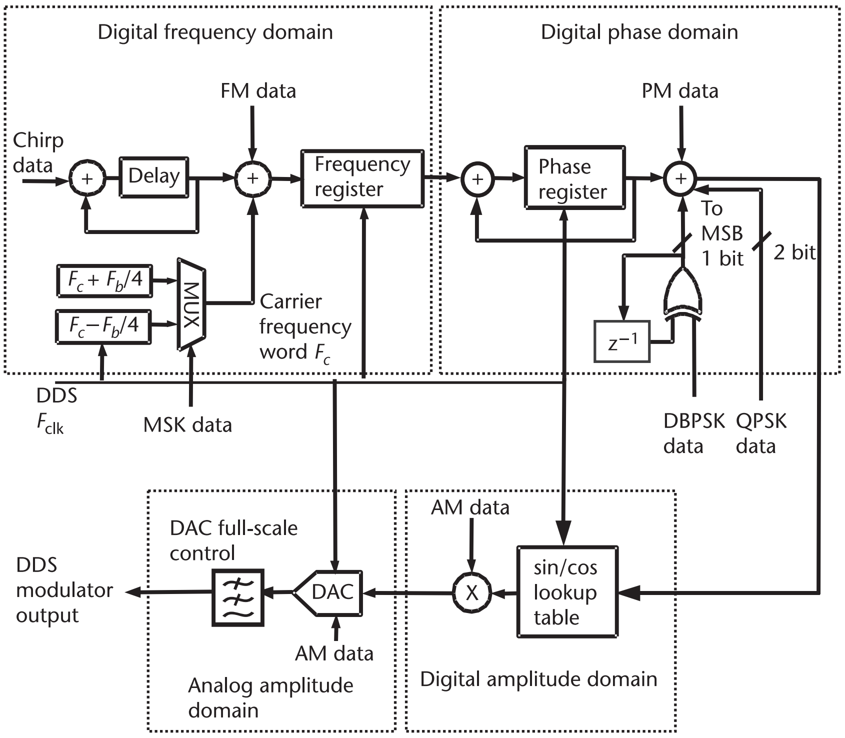


Figure 11.4 Direct modulation through a DDS.

noise, and faster switching, while the PLL can synthesize the carrier frequency at multigigahertz frequencies. It should be pointed out that all of the digital modulation implemented in DDS is accurate up to the limit set by the quantization levels due to the finite number of frequency, phase, and amplitude bits, while the accuracy of analog modulation is subject to the variation, noise, and nonlinearity of the analog circuits. The DDS synthesizer can implement types of modulation and waveforms, such as chirp, step frequency, frequency shift keying (FSK), FM, MSK, PM, AM, QAM, and other hybrid modulations, as shown in Figure 11.4. Next, we discuss the direct-modulation capability in a DDS.

11.3.1 Phase Modulation

Transmitting information requires more than just a carrier tone. The tone has to change in some way over time to indicate what information is being sent. In general, a carrier has two properties that can be changed or modulated in order to convey information to the receiver: amplitude and phase. We note that FM is not treated separately as it can be understood to be a subset of phase modulation. Some modulation schemes only change the amplitude of the signal, and some only change the phase (or frequency) of the signal, but some more complicated modulation schemes change both. We begin by considering modulation that changes only the phase of the carrier.

11.3.2 Phase Shift Keying

This section describes some of the basics of phase modulation. Then, towards the end of the section, the topic will return to the implementation of direct phase modulation of DDSs.

A phase shift keyed (PSK) modulated signal encodes data by changing the phase of the carrier signal according to which bits are to be transmitted. A PSK signal is given by

$$S_{\mathrm{PSK}}(t) = A \cdot \cos(\omega_{\mathrm{RF}} t + \phi_{\mathrm{bits}}) \tag{11.5}$$

where A is the amplitude of the carrier signal (a constant), ω_{RF} is the frequency of the carrier, and ϕ_{bits} is the phase, which is given by

$$\phi_{\mathrm{bits}} = \frac{2\pi}{2^N} \cdot (i - 1) \quad i = 1, 2, 3, \ldots, N \tag{11.6}$$

where $\log_2 N$ is the number of bits transmitted per phase change. For instance, if only one bit is transmitted per phase change, called binary phase shift keying (BPSK), then the phase is either 0° or 180° ($i = 1$ or 2), depending on whether a zero or a one is transmitted. If 2 bits are transmitted per phase change, called quadrature PSK (QPSK), then four phases are required. Similarly, 3 bits can be transmitted if eight phases are employed, and so on. So, why not use an infinite number of phases to get an infinite bit rate with this modulation scheme? The answer is that, in the presence of noise, the more phases that are used (therefore, the closer adjacent phases are to one another), the harder it is to determine one from another, and this increases the probability that an error will occur.

In general, the phase angle in PSK modulation can be represented as a vector consisting of I and Q components, as shown in Figure 11.5. A very important feature in this figure is that adjacent symbols differ by only one bit. This means that, if a phase is misinterpreted, a minimum number of bit errors will occur.

One additional refinement that is sometimes implemented with QPSK is to delay one of the baseband date streams (usually the Q path) by half the data rate. This means that the I and Q data never change at the same time. Thus, instead of getting a maximum possible instantaneous phase shift of 180°, the maximum instantaneous phase shift is only 90°. This makes the modulation more spectrally efficient. When this is done, the modulation is often referred to as *offset quadrature phase shift keying* (OQPSK).

In the configuration of an analog quadrature modulator, the baseband data is modulated onto a carrier with the aid of two mixers and an RF LO that generates two tones that are 90° apart, as shown in Figure 11.6 [11]. The data is first converted into I and Q phase components in the baseband. Then, that data is mixed to RF using two references that are phase shifted by 90°. When these two signals are added together, an RF carrier with the appropriate phase is generated.

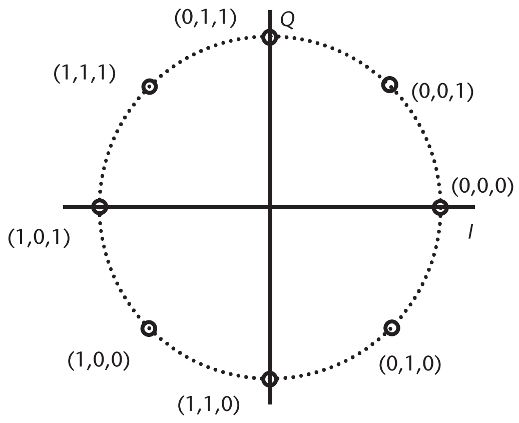


Figure 11.5 Phase plot of an 8-PSK modulated signal.

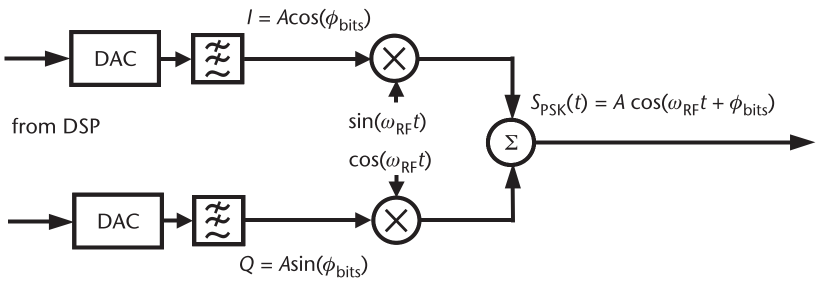


Figure 11.6 Basic quadrature modulator.

After the data has been transmitted, a receiver performs the inverse function to recover the transmitted information, as shown in Figure 11.7. The signal is mixed down from RF to baseband using two LOs that are separated by 90°. Assuming lowpass filtering and that the phase of the transmitted signal can be recovered, this gives back the I and Q data that was transmitted. This data is then fed into an analog-to-digital converter (ADC), and, from there, the DSP determines what phase was transmitted. Note that in the special case of BPSK, no Q path is needed, as there is only an I component at 0° or 180°. However, if the carrier frequency and phase are not recovered and exactly synchronized, there can be a relative phase shift, in which case both I and Q paths are required. Note that a receiver that recovers the phase of the transmitted signal is known as a coherent receiver. Receivers that do not recover the transmitted phase are noncoherent receivers.

Beyond determining the basic functional parts of different PSK transceivers, it is necessary to determine the required SNR for a given probability of a symbol error. This allows the completion of the system specifications. As a starting point, a given phase is transmitted, and, in the course of that transmission, noise is added that will tend to change the phase of the transmitted signal. If the phase is changed enough, it will pass a threshold, and the symbol will be wrongly interpreted, resulting in a symbol error, as Figure 11.8 illustrates [12]. Here, the nominal transmitted phase is at 0° and is at the center of a normal probability density function. The threshold for misinterpreting a symbol is $\pm\pi/N$ (the shaded area indicates when an error has been made). If the noise changes the phase by more than this amount, the symbol will be wrongly interpreted. Note that as long as only one bit changes when an adjacent phase is wrongly interpreted, the bit error rate as a function of the symbol error rate is given by

$$P_b \approx \frac{P_s}{\log_2 N} \tag{11.7}$$

A plot of the probability of symbol error versus E_b/N_0 can be derived based on statistics and the above discussion. An approximate formula for the probability of symbol error as a function of E_b/N_0 is given by [13]

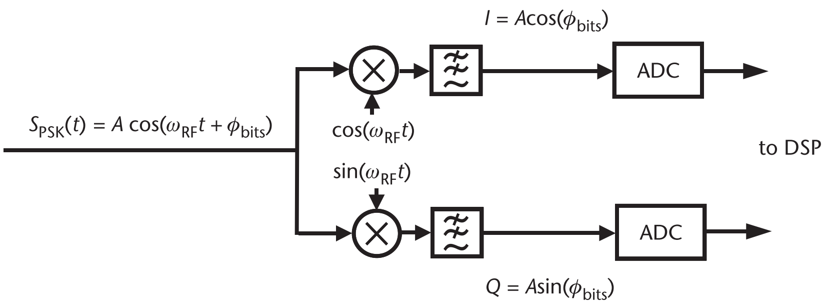

$I = A\cos(\phi_{\text{bits}})$
$s_{\text{PSK}}(t) = A\cos(\omega_{\text{RF}}t + \phi_{\text{bits}})$
$\cos(\omega_{\text{RF}}t)$
$\sin(\omega_{\text{RF}}t)$
ADC
ADC
to DSP
$Q = A\sin(\phi_{\text{bits}})$

Figure 11.7 Basic quadrature demodulator.

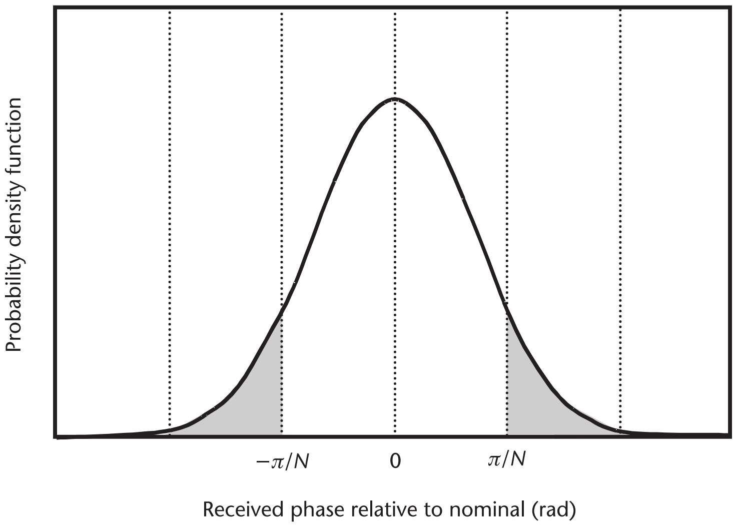


Figure 11.8 Probability density function of a PSK signal showing the probability of making a signal error.

$$P_s\left(\frac{E_b}{N_0}\right) \approx 2Q\left[\sqrt{2\frac{E_b}{N_0}\log_2(N)}\cdot\sin\left(\frac{\pi}{N}\right)\right] \tag{11.8}$$

where $Q(x)$ is the area underneath the tail of a Gaussian probability density function. $Q(x)$ is given by

$$Q(x) = \int_x^\infty \frac{1}{\sqrt{2\pi}} e^{-\frac{z^2}{2}} dz \tag{11.9}$$

The probability of symbol error is plotted in Figure 11.9. Obviously, as the number of symbols increases at a given E_b/N_0, the probability of error is larger.

Phase modulation can be implemented in a DDS, such as in Figure 11.4, by altering the phase word at the accumulator output prior to the ROM sin/cos lookup table. Note that the first MSB of the phase word represents 180° of phase weight, the second MSB represents 90° of phase weight, the third MSB represents 45° of phase weight, and so on. Therefore, various phase shifts can be implemented by toggling the corresponding phase bit. In general, for a P-bit phase word, its Lth bit counting from the right (LSB) has $180°/2^{P-L}$ of phase modulation weight, and the PM resolution is given by $360°/2^P$. For instance, a 10-bit phase word can be modulated with 0.3516° of phase resolution.

M-ary PSK (MPSK) can similarly be implemented by altering the $\log_2(M)$ MSBs of the phase word at the accumulator output. For instance, a BPSK can be implemented by toggling the MSB of the phase word using 1-bit baseband data, which causes 180° of phase transition. Data shaping using a Nyquist filter can be applied to the baseband data stream before the modulation. Differential BPSK (DBPSK) can also be implemented by XORing the baseband data with its delayed

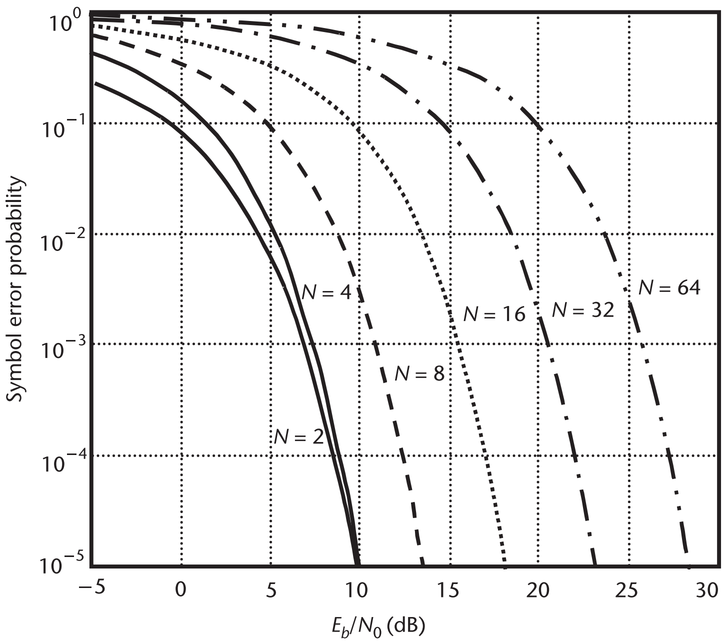


Figure 11.9 Probability of symbol error versus E_b/N_0 for PSK.

version, as shown in Figure 11.4. Similarly, QPSK can be implemented using 2-bit baseband QPSK data to modulate the two MSBs of the phase word, which generate four phases with 90° of resolution.

11.3.3 Frequency Modulation

In a PSK transceiver, the phase of the carrier is changed as a means to transmit information. On the other hand, a frequency shift keyed (FSK) modulated signal encodes data by changing the frequency of the carrier signal according to the particular bits that have been transmitted. An FSK signal is given by

$$S_{\mathrm{FSK}}(t) = A \cdot \cos[(\omega_{\mathrm{RF}} + \omega_i)t + \phi] \tag{11.10}$$

where A is the amplitude of the carrier signal (a constant), ω_{RF} is the nominal frequency of the carrier, ϕ is an arbitrary phase, and ω_i is the change in carrier frequency that determines what bits have been transmitted. Thus, in this modulation scheme, the RF waveform can be thought of as a set of different frequencies being turned on and off at the bit rate.

So what is to stop us from spacing the different frequencies corresponding to different bits infinitely close together and, thus, getting an infinite number of bits/sec/Hz? Remember that the signals (centered at their carrier frequencies) will have a power spectral density (PSD) similar to that shown in Figure 11.1. Thus, adjacent frequencies must be spaced so that there is minimal interference between different bits. A way to do this is to align the peak in the frequency response of 1 bit with the null in the frequency response of an adjacent bit, by spacing the frequencies either at the bit rate or at a multiple of the bit (or symbol) rate, as shown in Figure 11.10. Therefore,

$$\omega_i = f_B \cdot i \quad i = 1, 2, 3, \ldots, N \tag{11.11}$$

where $\log_2 N$ is the number of bits transmitted per frequency. Therefore, the maximum frequency deviation of the RF signal is

$$\Delta f_{\max} = f_B \cdot N \tag{11.12}$$

Thus, the required bandwidth is proportional to the number of bits that are transmitted simultaneously. This means that 2 bits transmitted simultaneously will take twice the bandwidth that 1 bit will take. As a result, the spectral efficiency of FSK drops off from around 0.5 bit/sec/Hz for binary frequency shift keying (BFSK) and 4-FSK to lower values as the number of bits increases. Note that even for BFSK, the spectral efficiency is half of that for BPSK. Therefore, FSK does not offer the same advantage in spectral efficiency as PSK. However, since all the frequencies are orthogonal, the presence of more frequencies does not affect the receiver's ability to detect any of them. As a result, higher data rates can be achieved without a reduction in the symbol error rate.

Figure 11.11 shows the probability of a symbol error versus E_b/N_0 for different numbers of bits per symbol for FSK. Note that the probability of error actually drops for a higher number of bits per symbol, but also note that this is because a higher bandwidth per bit is being used. Thus, for higher order FSK, the SNR will need to be higher to get the same E_b/N_0 ratio, as the bandwidth of the radio must be larger. This is different from PSK, where the ratio of bandwidth to the number of bits per symbol is relatively constant, regardless of the number of bits per symbol being transmitted. The probability of symbol error for FSK is given by [13]

$$P_s\left(\frac{E_b}{N_0}\right) \leq (N-1) \cdot Q\left[\sqrt{\frac{E_b}{N_0}\log_2(N)}\right] \tag{11.13}$$

In FSK, the signal occupies much more bandwidth than it does in the PSK case and is proportional to the number of bits per symbol being transmitted. As a result,

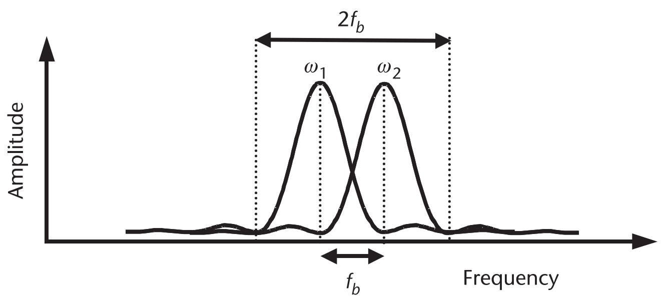


Figure 11.10 RF PSD for binary FSK.

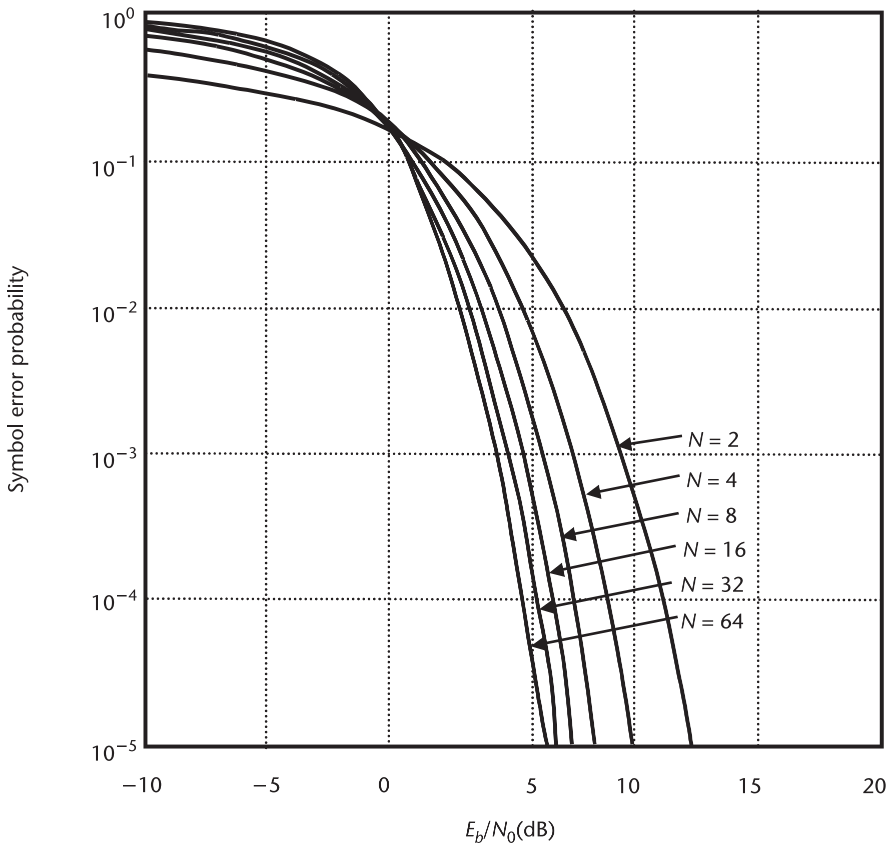


Figure 11.11 Probability of symbol error versus E_b/N_0 for FSK.

$$\frac{E_b}{N_0} = \frac{S}{N} \cdot \frac{\mathrm{BW}}{f_B \cdot \log_2 N} \approx \frac{S}{N} \cdot \frac{1}{\log_2 N} \tag{11.14}$$

An advantage of FSK over PSK is the simplicity of the transmitters and receivers. Figure 11.12 shows a basic FSK modulator. Note here that the DAC can directly drive the input to a VCO, resulting in an FM output. Note that FSK has constant amplitude and, therefore, does not require a linear power amplifier, making it possible to use a highly efficient, nonlinear power amplifier in the transmitter. Figure 11.13 shows a demodulator for FSK. Here, the signal is first passed through a limiter to make the output's amplitude constant. Then, a frequency discriminator is used to produce a voltage that is proportional to the frequency of the received

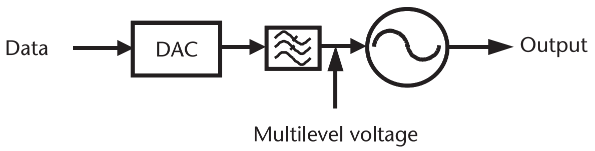


Figure 11.12 Basic FSK modulator.

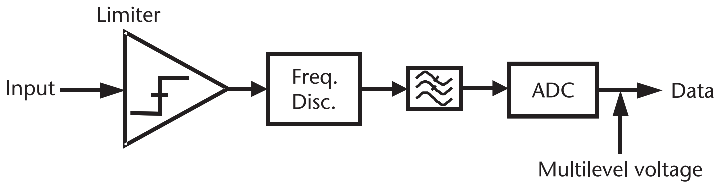


Figure 11.13 Basic FSK demodulator.

signal. This signal is then passed through an ADC. The output from the ADC is then used to determine what bits were sent at the transmitter.

Having discussed the basics of FM, we now turn back to modulation of the DDS. FM can be implemented by adding the baseband FM data to the carrier frequency word, as shown in Figure 11.4. In a DDS, FM generation is accomplished discretely. The frequency control word (FCW) is analogous to the dc, or slowly varying, tuning voltage of an analog VCO. The DDS has an adder that allows a digital offset adjustment of the center frequency. This offset is analogous to the analog-modulation input, an ac component, to the VCO tune line and could be from any baseband digital source, such as a digital voice, video, or data generator. If the center frequency word is F_c and the baseband FM data is represented by $F_b(j)$, then the instantaneous phase of the FM signal at the nth time step can be written as

$$P(nT) = T\sum_{j=1}^{n} [F_c + F_b(j)] + P_0 = F_c T + T\sum_{j=1}^{n} F_b(j) + P_0 \tag{11.15}$$

where P is the phase accumulator output, and P_0 is defined as the initial seed of the accumulator. T is the sample interval of the DDS clock, $T = 1/F_{\text{CLK}}$. This is analogous to the analog phase angle given by

$$\theta(t) = \omega_c t + K_f \int_0^1 f(\tau)\, d\tau + \theta_0 \tag{11.16}$$

The discrete counterparts of phase θ and time t are P and T, respectively. An accumulator overflow indicates that the phase has completed one revolution of 2π radians. The smallest increment of the discrete phase P is $1/2^N$, where N is the number of the phase accumulator bits. The DDS output frequency is the derivative of the phase with respect to the time:

$$F_{\text{out}} = \frac{\Delta P}{\Delta T} = \frac{\dfrac{\text{FCW}}{2^N}}{\dfrac{1}{F_{\text{CLK}}}} = \frac{F_{\text{CLK}}(F_c + F_b)}{2^N} \tag{11.17}$$

The maximum output frequency due to the Nyquist limit is half the clock frequency, which is when FCW reaches $1/2^{N-1}$. The minimum output frequency can be 0 Hz, making DDS a dc-coupled modulator, which is not possible through PLL synthesizers. The maximum FM deviation can be calculated as

$$\Delta F_{\max} = F_{\text{CLK}}/2^{N-L} \tag{11.18}$$

where L is the number of FM data bits. For example, if an FM signal is applied to a 1-GHz DDS system, and the MSB of the FM data is located 3 bits below the full scale of the FCW, then the maximum FM deviation = 10 GHz/2^3 = 125 MHz. Similarly, the frequency resolution of the FM signal is given by

$$\Delta F_{\min} = F_{\text{CLK}}/2^{N} \tag{11.19}$$

In a practical application, the phase noise of the clock overwhelms the round-off noise of the DDS phase accumulator. Note that a DDS involves a frequency downconversion process. The FM output signal will exhibit improved phase noise performance over the DDS clock by

$$\Phi_n = 20\ \log(F_{\text{CLK}}/F_{\text{OUT}})\ [\text{dB}] \tag{11.20}$$

Assuming a 1-GHz DDS clock, the FM output at 100 MHz will see a 20-dB SNR improvement over the clock. Temperature drift and jitter will also be reduced accordingly.

M-ary FSK (M-FSK) can be generated by toggling the FCW among a group of *M* carrier frequency words. Those frequency words can be programmed externally to allow any FSK modulation.

11.3.4 Minimum Shift Keying

A related modulation that can be thought of as either phase or frequency modulation is minimum shift keying (MSK). It is a very simple form of modulation that, during each bit period, either advances the phase of the carrier by 90° to indicate a one or retards the phase by 90° to indicate a zero. Thus, an MSK signal can be represented by

$$S_{\text{MSK}}(t) = A \cdot \cos\left[\omega_{\text{RF}} t + \frac{\pi}{2} \cdot d(t) \cdot t + \phi\right] \tag{11.21}$$

where $d(t)$ has a value of ± 1 to indicate the value of a bit that has been transmitted. In order to generate this phase change, the frequency must be instantaneously higher or lower than the carrier frequency over the bit period. Since the phase must change by $\pi/2$ in a time of T_B (the bit period), the frequency must be

$$\Delta f = \frac{\pi/2}{T_B} \cdot \frac{1}{2\pi} = \frac{f_B}{4} \tag{11.22}$$

higher or lower than the nominal carrier frequency for that bit period. Note that MSK is very similar to BFSK except that the two frequencies in MSK are spaced at half the separation of the tones in FSK modulation. This means that MSK is more spectrally efficient than BFSK. Also note that MSK is a form of continuous phase modulation. That means that there are no discontinuities in the phase of the transmitted waveform (FSK has no such restriction). Thus, either of the modulators shown in Figures 11.6 and 11.12 can be used with MSK; however, MSK will require the more complex demodulator shown in Figure 11.7 in order to be properly detected. This modulation has many properties similar to BPSK and QPSK and will have a bit error probability that is the same as QPSK as shown in Figure 11.8.

MSK can be easily implemented in a DDS, as shown in Figure 11.4. A unique technique to generate MSK in DDS is to use baseband MSK data to toggle two frequency words corresponding to $F_c \pm F_b/4$, where F_c is the carrier frequency, and F_b is the baseband MSK symbol data rate. When baseband MSK data is zero, the DDS FCW is given by $F_c - F_b/4$, which causes a continuous phase retardation of 90° within one bit period $1/F_b$. As a result, the DDS output can be written as

$$\begin{aligned} S_{\text{out}} &= \sin\left[2\pi\left(F_c \pm \frac{F_b}{4}\right)t\right] \qquad (11.23) \\ &= \cos\left(\pm\frac{\pi}{2}\cdot F_b t\right)\cdot \sin(2\pi\cdot F_c t) \pm \sin\left(\pm\frac{\pi}{2}\cdot F_b t\right)\cdot\cos(2\pi\cdot F_c t) \end{aligned}$$

This equation is equivalent to (11.21). As shown, the DDS output is an MSK signal, which can also be thought of as OQPSK with sinusoidal pulse shaping on the baseband signal. When baseband MSK data is one, the DDS FCW is given by $F_c + F_b/4$, which causes a continuous phase advance of 90° within one bit period $1/F_b$.

11.3.5 Step Frequency

A step frequency can be generated by toggling the FCW between two carrier frequency words. Unlike analog frequency modulation through a VCO in a PLL synthesizer, which requires a long settling time, frequency switching in a DDS can be settled within one clock cycle.

11.3.6 Chirp Waveforms

A chirp waveform can be generated by accumulating a fixed frequency chirp slope (chirp data in Figure 11.4). The frequency word at the accumulator output is thus a ramp signal. A DDS with such a linearly ramped frequency word as its input will perform linear frequency modulation, which generates a chirp waveform at the DDS output. The chirp waveform is used widely in test equipment, where swept frequency response of the device under test (DUT) is measured, and in radars and spread-spectrum communications, where the signal energy is spread across a wide bandwidth to reduce spectral density in multipath and fading environments. If nonlinear chirp is desired, a ROM that holds the desired frequency ramp information can be used before the phase accumulator. Hence, the frequency word of the DDS can vary based on any programmed chirp data.

11.3.7 Amplitude Modulation

Amplitude modulation can be done either in the digital domain or in the analog domain. In the digital domain, the amplitude word after the sin/cos lookup table in Figure 11.4 can be modulated using a digital multiplier, which is accurate up to the quantization level, limited by the finite number of amplitude bits. Alternatively, amplitude modulation can be implemented by adjusting the DAC full-scale current using the baseband amplitude modulation data, whose accuracy is limited by the nonlinearity and dynamic range of the DAC. The amplitude modulation in the digital domain is discrete, while the amplitude modulation in the analog domain is continuous.

11.3.8 Quadrature Amplitude Modulation

QAM can be thought of as an extension of QPSK. In QAM, the symbols are distinguished by having both different phases and different amplitudes. Thus, instead of four possible phases as in QPSK, a larger number of both phases and amplitudes are used to define which bit has been transmitted. Thus, rather than a constellation of four symbols, in QAM the constellation has 16, 64, 256, or more phase and amplitude locations corresponding to different bits being transmitted. Figure 11.14 shows the constellation for 16-QAM. In 16-QAM, 4 bits are transmitted simultaneously for a spectral efficiency of 4 bits/sec/Hz. Similarly, 64-QAM and 256-QAM achieve a spectral efficiency of 6 and 8 bits/sec/Hz, respectively.

QAM has an advantage over MPSK in that, for a given spectral efficiency, it will often achieve an equivalent bit error rate at a lower E_b/N_0. The probability of symbol error for QAM is given by [13]

$$P_s\left(\frac{E_b}{N_0}\right) \approx \frac{2(1 - L^{-1})}{\log_2 L} \cdot Q\left[\sqrt{\left(\frac{3 \log_2 (L)}{L^2 - 1}\right) \frac{2E_b}{N_0}}\right] \tag{11.24}$$

where $L = \sqrt{N}$ which is also the number of amplitude levels in one dimension. So, for instance, in the case of 64-QAM, $L = 16$. The symbol error probability is shown

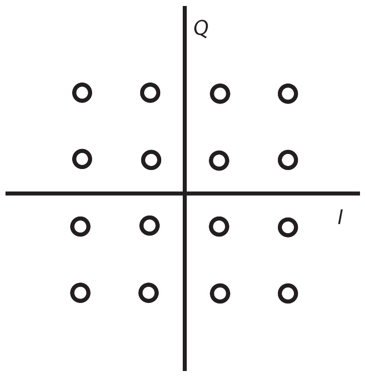


Figure 11.14 Phase plot of a 16-QAM modulated signal.

in Figure 11.15. This type of modulation can also use modulators and demodulators, such as those shown in Figures 11.6 and 11.7.

A DDS such as shown in Figure 11.4 can easily implement a QAM signal by simultaneously modulating its phase and amplitude words. Proper delay for the baseband amplitude data should be considered in order to synchronize the phase and amplitude modulation for the QAM signal.

11.3.9 Waveform Generation

In addition to direct modulation, various waveforms can also be generated through a DDS. In a DDS, arbitrary waveform data can be stored in a lookup ROM. The waveform data can also be programmable if the ROM is designed to be rewritable through external programming. For example, radar operational requirements include ever-increasing demands for affordable low-noise-signal and waveform generation. Radar sensitivity can be greatly increased if the signals are transmitted simultaneously through multiple carriers. In order to distinguish the received signals, carriers should differ from one another by frequency, phase, or the shape of the waveforms. A multifrequency transmission scheme is not favored since it increases transmission bandwidth and transceiver complexity. In contrast, multiphase or multiwaveform transmission is highly desirable since there is no bandwidth penalty. Ultimately, radar applications can benefit from the capabilities of DDS to provide arbitrary waveform generation.

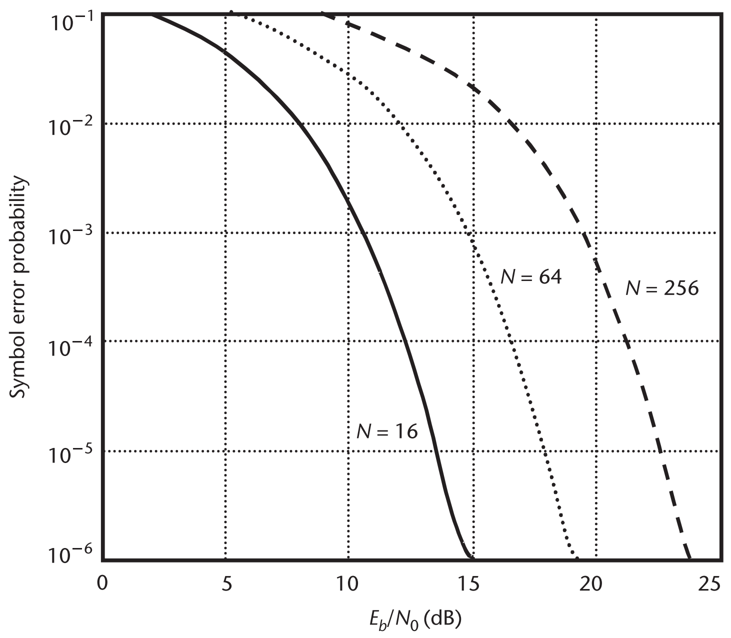


Figure 11.15 Probability of symbol error versus E_b/N_0 for QAM.

References

[1] Van Rooyen, G. J., and J. G. Lourens, "A Quadrature Baseband Approach to Direct Digital FM Synthesis," *IEEE Trans. on Broadcasting*, Vol. 46, No. 3, September 2000, pp. 227–230.

[2] Tan, L. K., and H. Samueli, "A 200 MHz Quadrature Digital Synthesizer/Mixer in 0.8 μm CMOS," *IEEE J. of Solid-State Circuits*, Vol. 30, No. 3, March 1995, pp. 193–200.

[3] Filiol, N. M., et al., "An Agile ISM Band Frequency Synthesizer with Built-In GMSK Data Modulation," *IEEE J. of Solid-State Circuits*, Vol. 33, July 1998, pp. 998–1008.

[4] Perrott, M. H., T. L. Tewksbury, and C. G. Sodini, "A 27-mW CMOS Fractional-*N* Synthesizer Using Digital Compensation for 2.5-Mbis GFSK Modulation," *IEEE J. of Solid-State Circuits*, Vol. 32, No. 12, December 1997, pp. 2048–2060.

[5] Twitchell, E. R., "A Digital Approach to an FM Exciter," *IEEE Trans. on Broadcasting*, Vol. 8, No. 2, June 1992, pp. 106–110.

[6] Goldberg, B. G., *Digital Frequency Synthesis Demystified: DDS and Fractional-N PLLs*, Eagle Rock, VA: LlH Technology Publishing, 1999.

[7] McMahill, D. R., and C. G. Sodmi, "A 2.5 Mbis GFSK 5.0 Mbis 4-FSK Automatically Calibrated DS Frequency Synthesizer," *IEEE J. of Solid-State Circuits*, Vol. 49, No. 5, January 2002, pp. 18–26.

[8] McMahill, D. R., and C. G. Sodini, "Automatic Calibration of Modulated Frequency Synthesizers," *IEEE Trans. on Circuits and Systems II: Analog and Digital Signal Processing*, Vol. 49, No. 5, May 2002, pp. 301–311.

[9] Perrott, M. H., M. D. Trott, and C. G. Sodini, "A Modeling Approach for Sigma-Delta Fractional-*N* Frequency Synthesizers Allowing Straightforward Noise Analysis," *IEEE J. of Solid-State Circuits*, Vol. 37, No. 8, August 2002, pp. 1028–1038.

[10] Jackson, T., G. Eapen, and F. Dai, "Linearized Offset QPSK Modulation Utilizing a Sigma-Delta Based Frequency Modulator," U.S. Patent Application Publication, No. 2002/0067773 A1, June 6, 2002.

[11] Larson, L. E., (ed.), *RF and Microwave Circuit Design for Wireless Communications*, Norwood, MA: Artech House, 1997.

[12] Proakis, J. G., *Digital Communications*, 3rd ed., New York: McGraw-Hill, 1995.

[13] Sklar, B., *Digital Communications: Fundamentals and Applications*, 2nd ed., Upper Saddle River, NJ: Prentice Hall, 2001.

APPENDIX A

A Review of Basic Control Theory

A.1 Introduction

Control systems are ubiquitous in modern society. They are everywhere and part of just about everyone's life, even if only engineers bother to describe them using equations. The heating system in your house is an example of a feedback system. You request a temperature in your house, and if the temperature falls below this value, the thermostat triggers the furnace to come on and increase the temperature. This is an example of negative feedback (the system takes action to keep the system output at constant level). This system is usually stable and is a well-understood example of a feedback-control system. The stock market is another example of a feedback system. If a stock price falls, triggering more selling and causing the stock price's decline to accelerate, this is an example of positive feedback. If however, a falling stock price encourages people to buy (perhaps they feel the stock is now at a bargain price), this is an example of negative feedback as this will probably cause the stock price to stabilize or rise again. Sadly, there is no equation in this appendix that will predict the stock market, and if the authors knew of one, they would be keeping it to themselves. This system is too complex and has too many inputs to be modeled with the techniques shown here. However, many engineering systems can be modeled using some math, and system behavior can be predicted quite well with control theory.

This appendix is intended to be a quick review of the control theory that most engineers study in undergraduate courses. However, such material is typically scattered over a few courses, and the intent here is to put all the concepts together in one location. Also, although continuous-time control theory is widely known, some people may be less familiar with methods of dealing with discrete-time systems.

Control theory is a facet of engineering that transcends specialties. For example, it is used nearly equally by radio frequency integrated circuit designers and civil and chemical engineers. The only real difference is in the specifics of the control system with which they are dealing. In electrical engineering, it is usually possible to "engineer" a system to have a desired behavior, and we need not be at the mercy of a system for which performance parameters are fixed by nature rather than the designer. Whatever the case, it is important to understand how feedback systems work, how to determine if they are stable, and how to determine more subtle aspects of their behavior.

A synthesizer has both digital and analog circuits. The phase detector and the dividers/counters are digital blocks. The loop filter and oscillator are analog blocks.

The charge pump has a digital input and an analog output, so it can be seen as the interface between the digital and the analog worlds. In order to determine the overall frequency response, delay, switching speed, stability, noise transfer functions, and so forth, we need to be able to combine the analysis of both the digital and the analog circuits. For this reason, we spend some time here describing the function of the digital circuit and developing analysis techniques.

A.2 The Continuous-Time Laplace Transform

Often, engineering systems need to be described by systems of differential equations. For example, the relationship between the voltage and current in a capacitor is

$$v(t) = \frac{1}{C}\int_0^t i(t)\,dt + \text{initial condition} \tag{A.1}$$

The hard way to do circuit design under deadline pressure is to perform all these integrations and differentiations directly. It would be far more convenient, therefore, smarter and faster, to apply a mathematical transformation that would permit very easy integration and differentiation of various components in a large system of differential equations, and, in fact, the Laplace transformation does just that.

For our purposes, the Laplace transform is used to transform a time domain waveform $f(t)$ into the s domain (complex frequency domain) $F(s)$. Once transformed, the new function is in a convenient domain in which to perform operations that involve integration or differentiation as these operations become algebraic in nature.

The s domain will be where we analyze a system's frequency response, and it will provide a convenient way for us to get the output of a system for a given input. The Laplace transform gives information about the frequency content of the waveform in terms of both amplitude and phase. The Laplace transform can be defined as [1]

$$F(s) = \int_0^\infty e^{-st} f(t)\,dt \tag{A.2}$$

The definition is given here for completeness. Most common functions that we will need are well known, and it is far easier to look them up in a table like Table A.1 than it is to solve the integral [1–3].

A.3 The Laplace Transform and Sampling

If the system of interest is a mixed-signal system (it uses both digital and analog parts), then at some point, the system's analog signals must be sampled. Sampling

Table A.1 Common Laplace Transforms and z Transforms

Time Domain Function $y(t)$/Discrete Time Domain $y(kT)$	*$F(s)$ (Continuous Laplace Equivalent)*	*$F(z)$ (Discrete Laplace Equivalent)*	*Comments and Additional Explanations*
$\delta(t)$ $\delta(kT)$	1	1	Unit impulse function
$u(t)$ $u(kT)$	$\frac{1}{s}$	$\frac{z}{z-1}$	Unit step function; also an integrator
t kT	$\frac{1}{s^2}$	$\frac{Tz}{(z-1)^2}$	Ramp
t^2 $(kT)^2$	$\frac{2}{s^3}$	$\frac{T^2z(z+1)}{(z-1)^3}$	Parabolic
t^n $(kT)^n$	$\frac{n!}{s^{n+1}}$	$\lim_{a\to 0}(-1)^n\frac{d^n}{da^n}\left(\frac{z}{z-e^{-aT}}\right)$	Exponential
e^{-aT} e^{-akT}	$\frac{1}{s+a}$	$\frac{z}{z-e^{-aT}}$	Natural exponential decay
$1-e^{-aT}$ $1-e^{-akT}$	$\frac{1}{s(s+a)}$	$\frac{z(1-e^{-aT})}{(z-e^{-aT})(z-1)}$	Natural exponential growth
$\sin(at)$ $\sin(akT)$	$\frac{a}{s^2+a^2}$	$\frac{z\sin(\omega T)}{z^2-2z\cos(\omega T)+1}$	Sine wave, no decay
$\cos(at)$ $\cos(akT)$	$\frac{s}{s^2+a^2}$	$\frac{z[z-\cos(\omega T)]}{z^2-2z\cos(\omega T)+1}$	Cosine wave, no decay
$e^{at}\sin(bt)$ $e^{akT}\sin(bkT)$	$\frac{b}{(s-a)^2+b^2}$	$\frac{ze^{-aT}\sin(\omega T)}{z^2-2ze^{-aT}\cos(\omega T)+e^{-aT}}$	Sine wave with exponential decay/ growth
$e^{at}\cos(bt)$ $e^{akT}\cos(bkT)$	$\frac{s-a}{(s-a)^2+b^2}$	$\frac{z^2-ze^{-aT}\cos(\omega T)}{z^2-2ze^{-aT}\cos(\omega T)+e^{-aT}}$	Cosine wave with exponential decay/ growth
$t\cdot e^{at}$ $kt\cdot e^{akT}$	$\frac{1}{(s-a)^2}$	$\frac{Te^{-aT}z}{(e^{-aT}z-1)^2}$	Ramp with exponential decay, growth
$e^{at}f(t)$ $e^{akT}f(kT)$	$F(s-a)$	$F(e^{-aT}z)$	Frequency-Shift Theorem
$f(t-nT)$ $f(kT-nT)$	$e^{-snT}F(s)$	$z^{-n}F(z)$	Time-Shift Theorem (a delay)
$\frac{df(t)}{dt}$ $f(kT)-f(kT-T)$	$sF(s)$	$(1-z^{-1})F(z)$	Differentiation Theorem/Difference Theorem
$\int_0^t f(\tau)\,d\tau$ $f(kT)+y(kT-T)$	$\frac{F(s)}{s}$	$\left(\frac{z}{z-1}\right)F(z)$	Integration Theorem/ Accumulation Theorem
$f(\infty)$	$\lim_{s\to 0} sF(s)$	$\lim_{z\to 1}(1-z^{-1})F(z)$	Final Value Theorem
$f(0)$	$\lim_{s\to\infty} sF(s)$	$\lim_{z\to\infty} F(z)$	Initial Value Theorem

Table A.1 (Continued)

Time Domain Function $y(t)$/Discrete Time Domain $y(kT)$	*$F(s)$ (Continuous Laplace Equivalent)*	*$F(z)$ (Discrete Laplace Equivalent)*	*Comments and Additional Explanations*
$f(t)g(t)$ $f(kT)g(kT)$	$F(s) \otimes G(s)$	$F(z) \otimes G(z)$	Convolution Theorem*
$\int_0^t f(t-\tau)g(\tau)\,d\tau = f(t) \otimes g(t)$ $f(kT) \otimes g(kT)$	$F(s)G(s)$	$F(z)G(z)$	Convolution Theorem*

*Note that the symbol ⊗ denotes convolution.

is the act of measuring a system's output periodically. Thus, only the system's outputs at the instances when sampling occurs are of interest, and anything the system does in between sampling instances is lost. Sampling is done once per clock cycle, then, the sampled value is held until the next sampling instance. In order to have a good digital representation of the waveform, it must be sampled fast enough. In fact, the waveform must be sampled at a rate that is at least twice as fast as the highest frequency of interest. This is known as the Nyquist sampling rate [4]. A sampler also causes delay; therefore, it can also be thought of as a delay block. An example of a delay block is a D-flip-flop where the input signal is read on the edge of the clock while, at the same time, the signal stored on the flip-flop (the previous input) is passed on to the next circuit. A similar effect can be seen in a sample-and-hold, such as the one in Figure A.1, where an input analog voltage is sampled, and the voltage is stored on a capacitor. During the hold phase, an additional circuit can take a sample of this voltage. Thus, this second sample still represents the input voltage except that it is delayed by some period. This is shown in Figure A.2. Note also that it is highly desirable to make sure that the two switches are never closed at the same time because this can result in sampling errors.

Thus, samples are taken at the points indicated by the circles. In this example, two sample-and-holds have been used to convert the continuous analog waveform into a sampled analog waveform. After the second sample-and-hold, the sample is ready to be passed on at the points indicated by the squares. The output clearly has the same voltage level as the input at the sampling instants. The output now has a staircaselike waveform, with output samples delayed by a full clock period

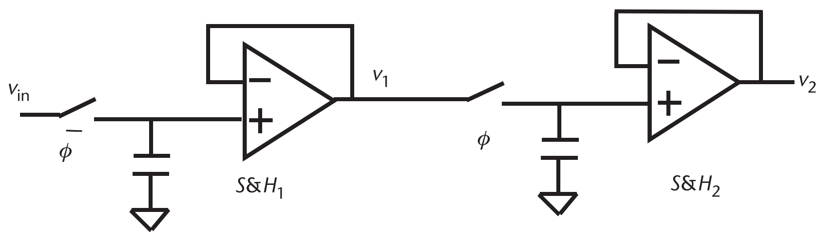


Figure A.1 Sample-and-hold circuit.

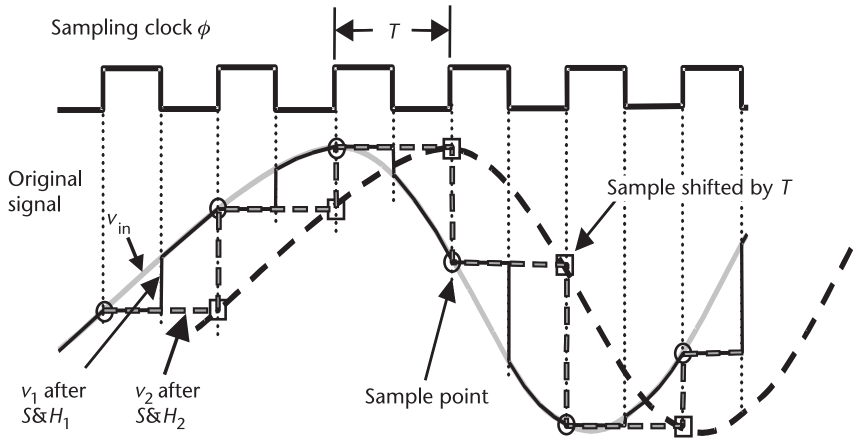


Figure A.2 Waveforms in a sample-and-hold circuit.

from the time the samples were taken. The mathematics of such a delay leads to the z transform.

We will now describe the process of sampling in more mathematical terms. Consider the continuous-time waveform in Figure A.3. Sampling can be described mathematically as multiplying the waveform by a series of delta functions, sometimes called an impulse train, as shown in Figure A.3. The waveform $f(t)$ sampled every T time units can be written as

$$f_s(t) = f(t)\sum_{k=0}^{\infty}\delta(t-kT) = \sum_{k=0}^{\infty} f(kT)\,\delta(t-kT) \tag{A.3}$$

We can find the Laplace transform of (A.3) by making use of Table A.1.

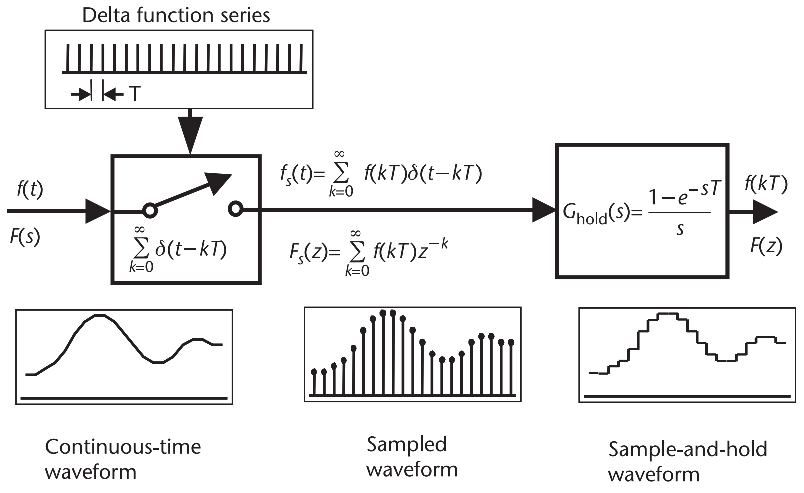

Delta function series
T
$f(t)$
$F(s)$
$\sum_{k=0}^{\infty}\delta(t-kT)$
$f_s(t)=\sum_{k=0}^{\infty} f(kT)\delta(t-kT)$
$F_s(z)=\sum_{k=0}^{\infty} f(kT)z^{-k}$
$G_{hold}(s)=\frac{1-e^{-sT}}{s}$
$f(kT)$
$F(z)$
Continuous-time waveform
Sampled waveform
Sample-and-hold waveform

Figure A.3 Illustration of sampling a waveform.

$$F_s(s) = F(s) \otimes \sum_{k=0}^{\infty} e^{-skT} = \sum_{k=0}^{\infty} f(kT)e^{-skT} \tag{A.4}$$

Thus, we can see that delay in the time domain is equivalent to phase shift in the frequency domain. This exponential term, which is the transform of the time delay, is used so frequently that it is now commonly replaced with z as follows:

$$z^{-1} = e^{-sT} \tag{A.5}$$

When this substitution is made, we then typically talk about z transforms:

$$F_s(z) = \sum_{k=0}^{\infty} f(kT)z^{-k} \tag{A.6}$$

Thus, z^{-k} can also be thought of as a delay of k clock periods. For example, if a system has an output that is not only dependent on the input but on past values of the input at times like $k - 1$, $k - 2$, $k - 3$, and so forth, in the time domain, then, in the frequency domain, this results in multiplication by z^{-1}, z^{-2}, z^{-3}, and so forth. In other words, a circuit block that provides a unit clock delay can be represented by a frequency domain block with a transfer function of z^{-1}.

Recall that in the earlier discussion of the sample-and-hold circuit, the discrete-time output signal was delayed by one sample clock period from the input signal. Thus, at low frequencies compared to the sampling frequency, the sample-and-hold behaves as a unit time delay element that can be described as z^{-1}. Furthermore, as will be shown in other parts of this book, we can easily use this math to describe fully the behavior of circuits that are built up of delay elements with summers and connected by feedback paths. This math is ideal for use in analysis of discrete-time circuits and systems, be they analog, sample-and-hold, or fully digital using register banks as delay elements.

Example A.1: Computing z Transforms

Illustrate the use of the preceding equations to take a time domain waveform and transform it to the z domain.

Solution: For example, the unit step function $u(t)$ can be transformed. In this case, $f(kT) = 1$ for all $k > 0$. Therefore,

$$F_s(z) = \sum_{k=0}^{\infty} z^{-k} = (1 + z^{-1} + z^{-2} + z^{-3} \ldots) \tag{A.7}$$

We make use of the following mathematical identity:

$$\frac{x}{x-1} = 1 + x^{-1} + x^{-2} + x^{-3} \ldots \tag{A.8}$$

Therefore,

$$F_s(z) = \frac{z}{z-1} \tag{A.9}$$

Other functions can be transformed in a similar way. For example, the natural exponential $f(kT) = e^{-akT}$:

$$F_s(z) = \sum_{k=0}^{\infty} e^{-akT} z^{-k} = 1 + e^{-aT} z^{-1} + e^{-a2T} z^{-2} + e^{-a3T} z^{-3} \ldots \tag{A.10}$$

This can be rewritten as

$$F_s(z) = 1 + (e^{aT} z)^{-1} + (e^{aT} z)^{-2} + (e^{aT} z)^{-3} \ldots \tag{A.11}$$

Again using the math identity (A.8),

$$F_s(z) = \frac{e^{aT} z}{e^{aT} z - 1} = \frac{z}{z - e^{-aT}} \tag{A.12}$$

In this way, the third column in Table A.1 can be constructed.

So far, we have talked about ideal sampling (multiplying by impulse functions). In reality, we also "hold" the value for an entire clock period. Thus, each element in the infinite series in (A.6) is multiplied by a square wave of length of one period:

$$g_{\text{hold}}(t) = u(t) - u(t-T) \tag{A.13}$$

$$G_{\text{hold}}(s) = \frac{1 - e^{-sT}}{s}$$

Some care should be taken in general when manipulating and converting systems from the s to the z domains. Note especially that, in general,

$$Z[G(s)H(s)] \neq G(z)H(z) \tag{A.14}$$

A.4 System Modeling with Frequency Response

A great majority of the systems we are interested in can be described by an amplitude (or magnitude) response and a phase response, both of which change with frequency. That is, if an input is applied to a system (e.g., voltage, current, power, phase, grenade), the output will be at some magnitude relative to the input, as well as some phase shift relative to the input. Either the s or the z domain can be used to describe these systems very conveniently.

A.4.1 Frequency Response of Continuous Systems

The simplest possible system is a first-order one, which requires that s be raised only to the first power to describe it. More complicated systems need a higher-order equation to describe them. A general first-order transfer function (the ratio of the system output to input) for a system without a zero (the function does not equal zero for any finite value of s) is given by

$$F(s) = \frac{K}{s + \omega_o} \tag{A.15}$$

It can be seen that $F(s)$ will blow up to infinity for $s = -\omega_o$. Any value of s that causes the expression to go to infinity is called a pole of the system. A first-order system has one pole given by

$$P = -\omega_o \tag{A.16}$$

A second-order transfer function (again without zeros) is given by

$$G(s) = \frac{K}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{A.17}$$

where ω_n is called the natural frequency, and ζ is the damping constant. For ζ less than or equal to zero, this function can be shown to blow up to infinity for some positive complex value of s. These values for ζ will be left until the next section. In general, the poles of this transfer function are given by

$$P_{1,2} = -\zeta\omega_n \pm \omega_n\sqrt{\zeta^2 - 1} \tag{A.18}$$

In the case for which ζ is greater than one, the poles are both real and have no imaginary component. If ζ is less than one, then the poles have an imaginary component. Thus, we often classify a system by its value of ζ. Systems with ζ greater than one are called overdamped; systems with a ζ of one are called critically damped, and systems with ζ less than one are called underdamped.

Poles are often plotted in the complex plain with $s = \sigma + j\omega$, as shown in Figure A.4. This plot shows the locations of the poles and equations for calculating the positions of the poles, knowing the natural frequency of the system and the damping constant.

These functions can be plotted to give the amplitude and phase response of the system letting $s = j\omega$. For a first-order system, this is quite straightforward

$$|F(j\omega)| = \frac{K}{\sqrt{\omega^2 + \omega_o^2}} \tag{A.19}$$

$$\angle F(j\omega) = -\tan^{-1}\left(\frac{\omega}{\omega_o}\right) \tag{A.20}$$

For the second-order case, the situation is more complex. The magnitudes and phase shifts are given by

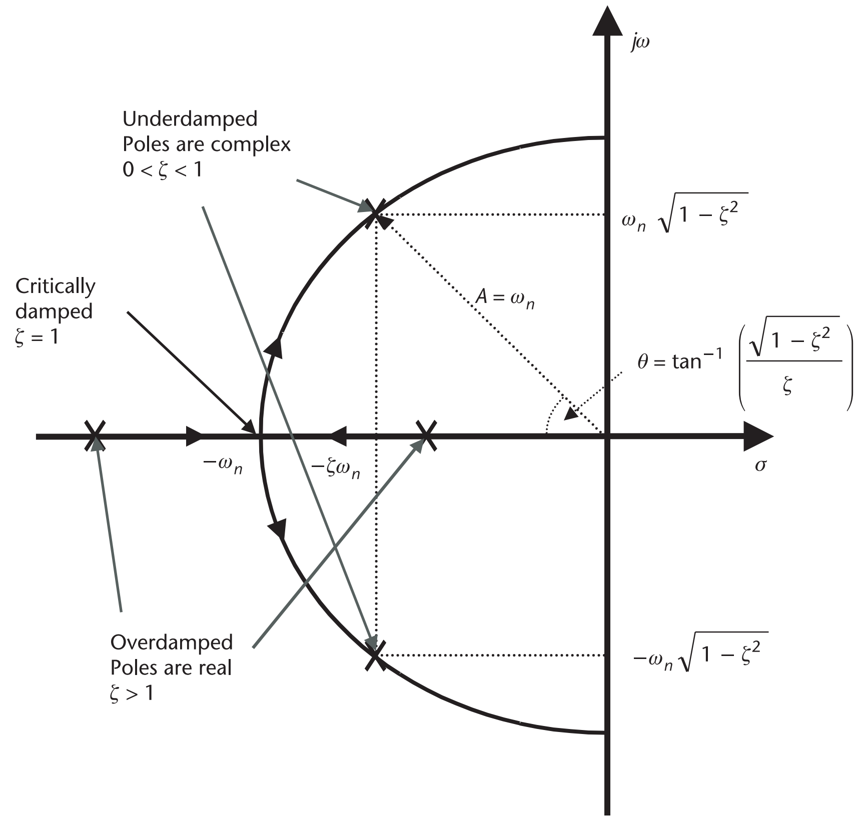


Figure A.4 Movement of second-order system poles plotted in the complex *s* plain for a sweep of ζ from $\zeta > 1$ to $\zeta = 0$.

$$|G(j\omega)| = \frac{K}{\sqrt{\left(\omega_n^2 - \omega^2\right)^2 + 4\zeta^2\omega^2\omega_n^2}} = \frac{K}{\sqrt{\omega^4 + \omega_n^4 + \omega_n^2\omega^2(4\zeta^2 - 2)}} \tag{A.21}$$

$$\angle G(j\omega) = \begin{cases} -\tan^{-1}\left(\dfrac{2\zeta\omega_n\omega}{\omega_n^2 - \omega^2}\right) & \text{for } \omega \le \omega_n \\ \pi - \tan^{-1}\left(\dfrac{2\zeta\omega_n\omega}{\omega_n^2 - \omega^2}\right) & \text{for } \omega > \omega_n \end{cases} \tag{A.22}$$

These expressions are plotted in Figures A.5 and A.6.

It is often of interest to know the 3-dB bandwidth of the system. This is the frequency at which the gain is 3 dB below the gain at dc. This can be found for the first-order system by setting (A.19) equal to

$$\frac{K}{\sqrt{\omega_{3\,\text{dB}}^2 + \omega_o^2}} = \frac{1}{\sqrt{2}} \cdot \frac{K}{\omega_o} \tag{A.23}$$

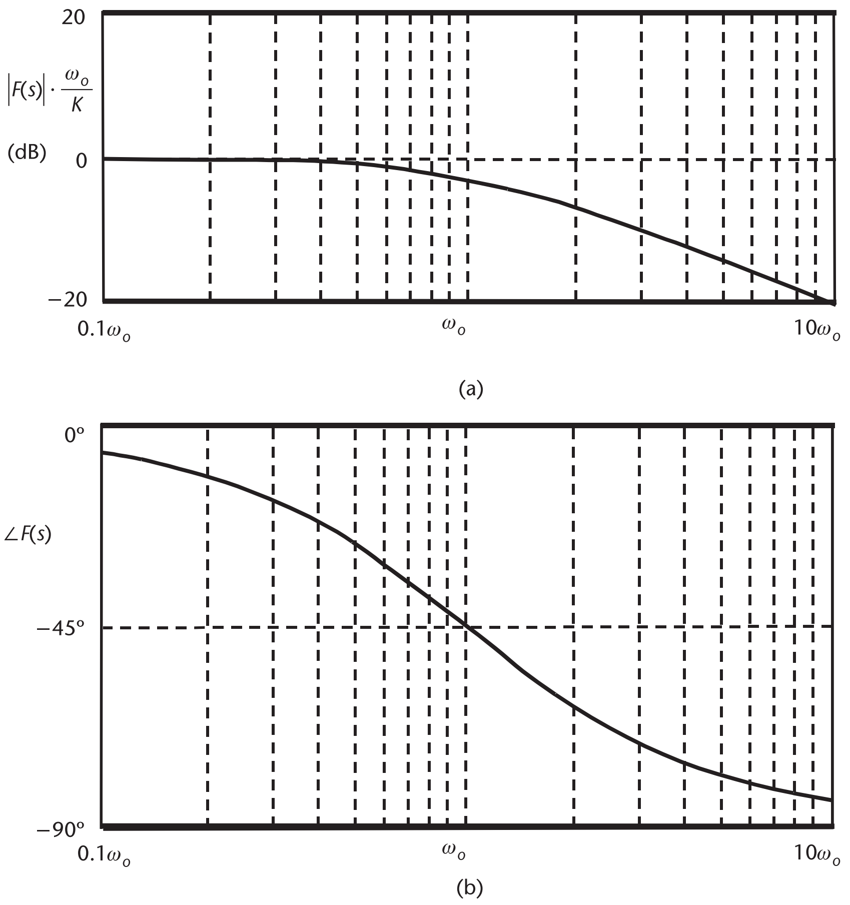


Figure A.5 (a) The amplitude, and (b) the phase of a first-order system.

where K/ω_o^2 is the dc value of this function. Solving this equation for $\omega_{3\text{ dB}}$ gives

$$\omega_{3\text{ dB}} = \omega_o \tag{A.24}$$

Similarly, for the second-order system [see (A.21)], which has a dc gain of K/ω_n^2,we can find this point as well:

$$\frac{K}{\sqrt{\omega_{3\text{ dB}}^4 + \omega_n^4 + \omega_n^2\omega_{3\text{ dB}}^2(4\zeta^2 - 2)}} = \frac{1}{\sqrt{2}} \cdot \frac{K}{\omega_n^2} \tag{A.25}$$

Solving this formula for $\omega_{3\text{ dB}}$ gives

$$\omega_{3\text{ dB}} = \omega_n\sqrt{1 - 2\zeta^2 + \sqrt{4\zeta^4 - 4\zeta^2 + 2}} \tag{A.26}$$

It should be noted that, although it is sometimes assumed that the 3-dB bandwidth of the system is approximately equal to the natural frequency, this is not

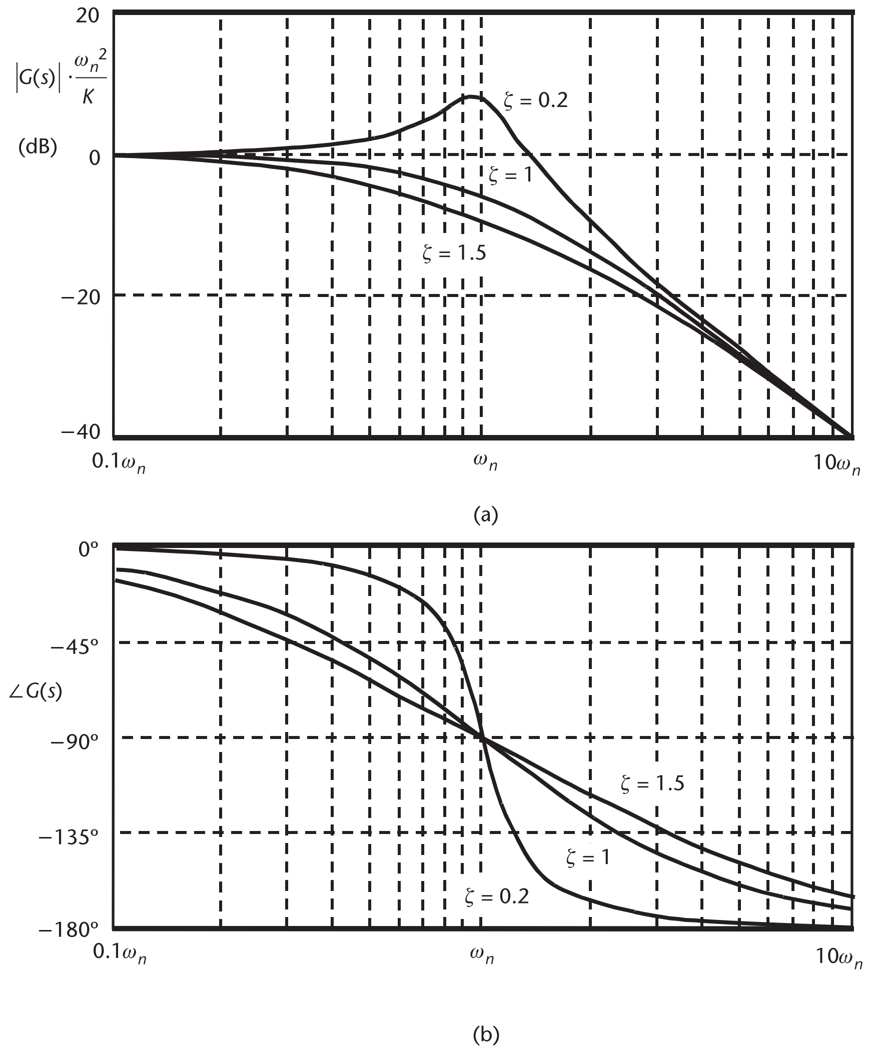


Figure A.6 (a) The amplitude, and (b) the phase of a second-order system for different values of ζ.

the case since, as can be seen by (A.26) and Figure A.6, the 3-dB bandwidth is actually dependent on the value of the damping constant.

In any system, there may also be one or two zeros present. Zeros can dramatically change the frequency response of the system in question. In general, a second-order system can be of the form

$$H(s) = \frac{K(s + \omega_{z1}) \cdot (s + \omega_{z2})}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{A.27}$$

If there is only one zero, and if its frequency is much higher than the pole frequency, then the amplitude plot will be nearly the same as that shown in Figure A.6. The difference will be that, after the zero frequency is reached, the roll-off rate will be –20 dB/dec rather than –40 dB/dec. Also, if the zero is in the left half of the s plane, then it will add phase lead of 90° at high frequencies; if it is in the right half of the s plane, then it will add 90° of phase lag at higher frequencies. If a zero is placed at the origin, then the gain at dc will drop to zero. For such a system, as frequency increases, gain will increase up to the first pole, in other words, resulting in a highpass frequency response. In general, any system for which the zero frequency is less than the pole frequencies will have a highpass response. If a pole and zero are at the same frequency, resulting in pole-zero cancellation, then the second-order system will behave like a first-order system.

A.4.2 Frequency Response of Sampled Systems

Even though we will primarily use the Laplace transform, in control theory there is a related transform known as the Fourier transform, which will give us direct information about the spectral content of the waveform. The Fourier transform is defined as [5]

$$F(2\pi f) = \int_{-\infty}^{\infty} e^{-j2\pi ft} f(t)\, dt \tag{A.28}$$

We note that the convolution theorem also applies to the Fourier transform, just as it does to the Laplace transform. The Fourier transform of the sampled waveform described by (A.3) is

$$F_s(f) = F(f) \otimes \frac{1}{T} \sum_{k=-\infty}^{\infty} \delta\left(f - \frac{k}{T}\right) \tag{A.29}$$

We note here, without proof, that the Fourier transform of a series of equally spaced delta functions in the time domain (to represent sampling, as in Figure A.3) is a set of delta functions in the frequency domain, as shown in Figure A.7. These are now convolved with the spectrum of the input waveform.

The effect of convolving the input waveform with a delta function is to recenter the waveform at the frequency of the delta function. These impulses can be seen to occur at dc, at the sampling frequency f_s, and at $2f_s$, $3f_s$, and so forth. Thus, the frequency spectrum of the continuous waveform repeats around f_s and around all multiples of the sampling frequency, as shown in Figure A.8. These multiple identical spectral components are the result of mixing the input with the sampling frequency. This mixing property of sampled systems is called *replication*. A consequence of replication is that signals fed into the system at frequencies close to the sampling frequency, or a multiple of the sampling frequency, will produce an

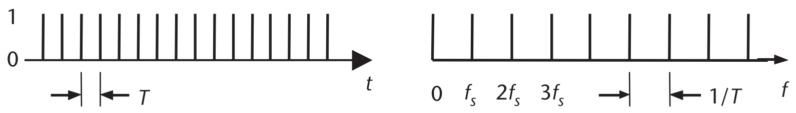


Figure A.7 Impulses in the time domain become frequency domain pulses.

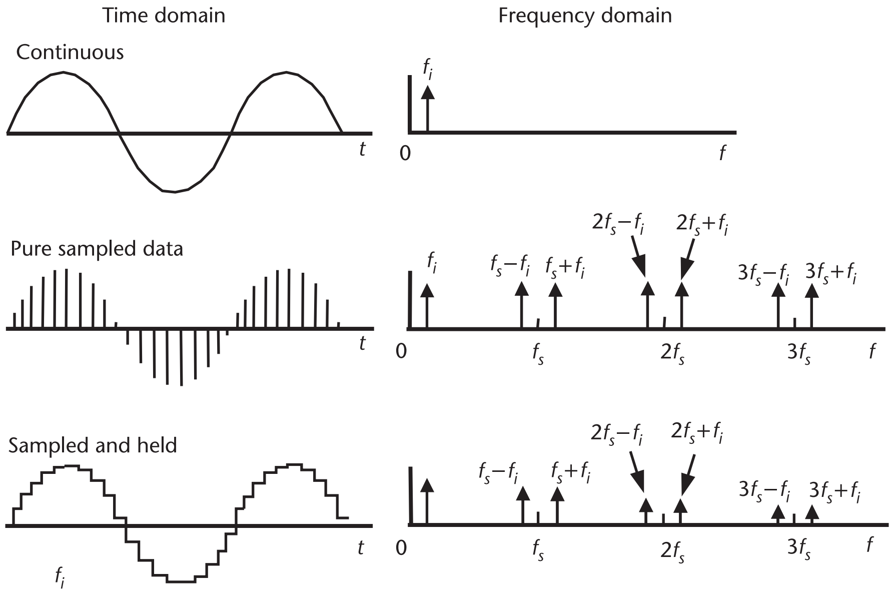


Figure A.8 The effect of replication on an input signal f_i: (a) continuous, (b) pure sampled data, and (c) sampled-and-held data.

output signal in the baseband (somewhere between dc and $f_s/2$). Such signals are indistinguishable from signals fed in close to dc. This typically unwanted output signal is called an aliased component.

If the frequency response of a continuous transfer function is known, the equivalent response in the sampled z domain can be determined. For instance, a sampled first-order system equivalent to the one described by (A.15) is given by

$$F(z) = \frac{Kz}{z - e^{-\omega_o T}} \tag{A.30}$$

In general, the frequency response could be determined by replacing z with e^{sT} and using a math program to solve this numerically. However, since this is equivalent to the continuous first-order system just considered, this will have a frequency response that repeats every $1/T$ Hz. This response will be shown later in Figure A.9.

When sampling is accompanied by a hold function, the frequency response is first convolved with the delta function to represent the sampling function. Then, the result is multiplied by the hold function. We use (A.13), replace s with $j\omega$, and multiply top and bottom by $e^{j\omega T/2}$ to result in

$$\begin{aligned} G_{\text{hold}}(j\omega) &= \frac{1 - e^{-j\omega T}}{j\omega} \frac{e^{j\omega T/2}}{e^{j\omega T/2}} \\ &= Te^{-j\omega T/2} \frac{e^{j\omega T/2} - e^{-j\omega T/2}}{2j\omega T/2} \\ &= Te^{-j\omega T/2} \frac{\sin(\omega T/2)}{(\omega T/2)} \end{aligned} \tag{A.31}$$

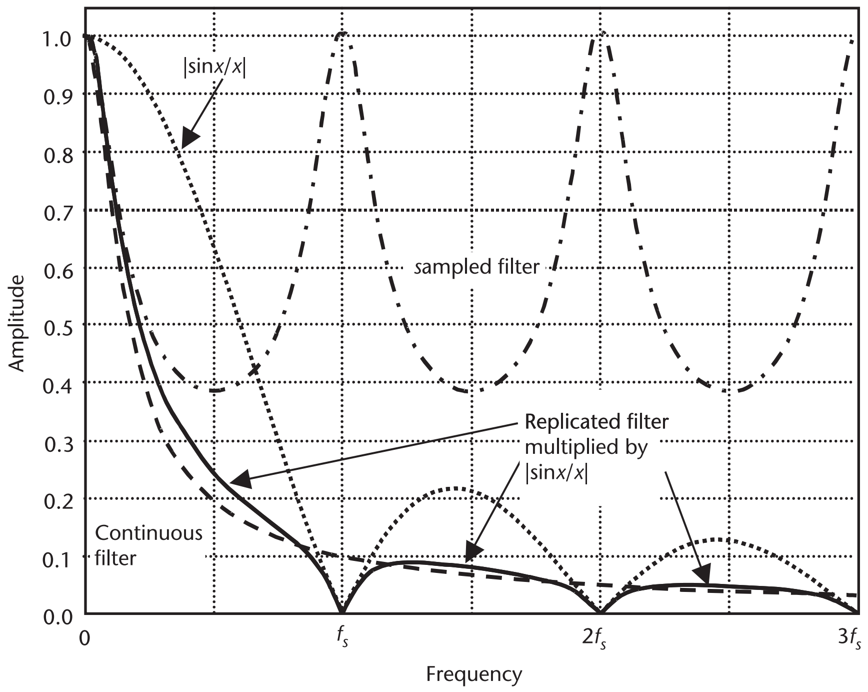


Figure A.9 Illustration of the difference between pure sampled data and sampled-and-held data.

Thus, there is a linear phase shift, and the amplitude is attenuated at higher frequencies. The result of the hold function is that successive replications become smaller and smaller, so the hold function is serving a very useful filtering function. This is illustrated in Figure A.8 for a signal, and the effects of impulse sampling and sample-and-hold on a transfer function are shown in Figure A.9.

It is worthwhile to note at this stage that when plotting the amplitude of impulse-sampled waveforms, the amplitude will not match that of the continuous-time waveform. This is because the average amplitude in an impulse sample is

$$A_{\text{ave}} = \frac{1}{T}\int f(kT)\,\delta(t)\,dt = \frac{f(kT)}{T} \tag{A.32}$$

However, after the hold function, the average amplitude in a sample is

$$A_{\text{ave}} = \frac{1}{T}\int f(kT)\,[u(kT) - u(kT+T)]\,dt = f(kT) \tag{A.33}$$

Note that the continuous-time waveform will also have an average amplitude of $f(kT)$ (provided its amplitude changes relatively little in the integration time T or, equivalently, its frequency is much lower than the sampling frequency). Thus, the amplitude of the impulse sampled waveform or transfer function is scaled by a factor of T relative to either the continuous-time or sampled-and-held waveform.

There are also some simple ways to do quick sketches of transfer functions. From (A.5), we can determine the effect of z on the phase shift of a system. For instance, the phase of $F(z) = z^{-1}$ as a function of frequency is

$$F(z) = z^{-1} = e^{-j2\pi\frac{f}{f_s}} = 1\angle - 2\pi\frac{f}{f_s} = \cos\left(2\pi\frac{f}{f_s}\right) - j\sin\left(2\pi\frac{f}{f_s}\right) \quad (A.34)$$

This is a vector with a magnitude of one and a phase of f/f_s as a fraction of 360°. Table A.2 summarizes this result.

Example A.2: Discrete and Continuous Integrators

Compare the phase of a simple integrator in the continuous-time domain to one in the discrete-time domain.

Solution: We first assume that the sampling frequency is normalized to one. We can look up the equations for continuous and discrete integrators in Table A.1. Then, using the formula $z = e^{sT}$ and knowing that $s = j\omega$, we can find the magnitude and angle of the two functions. These are shown for a few frequencies in Table A.3.

A comparison of the amplitude and phase responses of these two systems is shown in Figure A.10.

Here we can see the performance of the discrete integrator compared to the ideal continuous-time integrator. The figure shows that the expected continuous gain is halved for every doubling of frequency. This relationship is accurate for the discrete integrator up to about $f_s/8$ or even $f_s/4$. As for phase, the expected ideal response is a constant phase shift of 90°, while for the discrete integrator, the phase is close to 90° only for low frequencies.

A.5 Response in the Time Domain

There are three classic inputs to a system, which we usually consider when discussing transient response. Here, the transient response refers to the system's response to

Table A.2 Frequency Response of z^{-1}

Frequency f/f_s	$f = 0$	$f = f_s/8$	$f = f_s/4$	$f = f_s/2$	$f = f_s$
Vector Direction	→	↘	↓	←	→
z^{-1}, Cartesian form	1	$\frac{1}{\sqrt{2}} - j\frac{1}{\sqrt{2}}$	$-j$	-1	1
z^{-1}, Polar form	$1\angle 0°$	$1\angle -45°$	$1\angle -90°$	$1\angle 180°$	$1\angle 0°$

Table A.3 Frequency Response of a Continuous and Discrete Integrator

Frequency	0	$f_s/32$	$f_s/16$	$f_s/8$	$f_s/4$	$f_s/2$	f_s
$\frac{z}{z-1}$	$\infty\angle -90°$	$5.1\angle -84.4°$	$2.56\angle -78.8°$	$1.31\angle -67.6°$	$0.707\angle -45°$	$0.5\angle 0°$	$\infty\angle -90°$
$\frac{1}{s}$	$\infty\angle -90°$	$5.09\angle -90°$	$2.55\angle -90°$	$1.27\angle -90°$	$0.637\angle -90°$	$0.318\angle -90°$	$0.159\angle -90°$

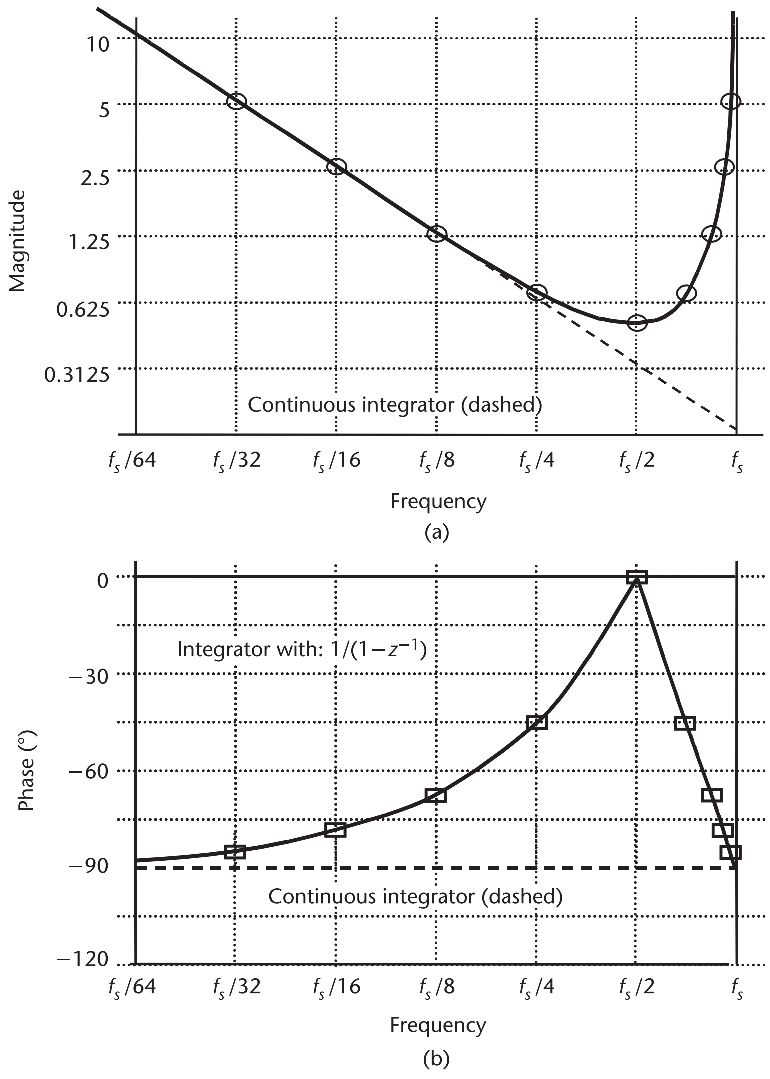


Figure A.10 (a) Magnitude, and (b) phase response of discrete integrator compared to a continuous integrator.

an impulse, a step, or a ramp. The response to an impulse is also sometimes called the natural response of the system. Since the impulse is instantaneous, there are no lasting effects on the system; so, what it does after the impulse has been applied is due to its own characteristics and properties only and not the properties of the input. Thus, to find the outputs of these systems, the system transfer function must be multiplied by an input of 1 for an impulse, an input of $1/s$ for a step, and an input of $1/s^2$ for a ramp, as given in Table A.1. To find the transient response, we simply take the inverse Laplace transform of the output. For a first-order system, the output equations are

$$f_{\text{impulse}}(t) = Ke^{-\omega_o t} \tag{A.35}$$

$$f_{\text{step}}(t) = \frac{K}{\omega_o} - \frac{K}{\omega_o} e^{-\omega_o t} \tag{A.36}$$

$$f_{\text{ramp}}(t) = \frac{K}{\omega_o} t - \frac{K}{\omega_o^2} + \frac{K}{\omega_o^2} e^{-\omega_o t} \tag{A.37}$$

Note that the above expressions assume a unit impulse, step, and ramp. In general, there could be a constant of proportionality associated with the expression as well.

The step is the most interesting. The response is an exponential rise having an initial value of zero and a final value of K/ω_o. The system has a time constant of value $1/\omega_o$, and if we say that the system has settled to its new value when it is 98% of the way there, then it settles in a time of

$$T_{\text{settling}} = \frac{\ln(0.02)}{\omega_o} \tag{A.38}$$

The second-order system is again slightly more complicated. In the case where ζ is greater than one, the poles are real, and the response is exponential in nature,

$$g_{\text{impulse}}(t) = \frac{K}{\omega_n\sqrt{\zeta^2 - 1}} e^{-\zeta\omega_n t} \sinh\left(\omega_n t\sqrt{\zeta^2 - 1}\right) \tag{A.39}$$

$$g_{\text{step}}(t) = \frac{K}{\omega_n^2}\left\{1 - e^{-\zeta\omega_n t}\left[\cosh\left(\omega_n t\sqrt{\zeta^2 - 1}\right) + \left(\frac{\zeta}{\sqrt{\zeta^2 - 1}}\right)\sinh\left(\omega_n t\sqrt{\zeta^2 - 1}\right)\right]\right\} \tag{A.40}$$

In the case where ζ is equal to one,

$$g_{\text{impulse}}(t) = \frac{Kt}{2} e^{-\omega_n t} \tag{A.41}$$

$$g_{\text{step}}(t) = \frac{K}{\omega_n^2}\left(1 - e^{-\omega_n t} - \frac{\omega_n t}{2} e^{-\omega_n t}\right) \tag{A.42}$$

Next, we will take the case where ζ is less than one. In this case the response is

$$g_{\text{impulse}}(t) = \frac{K}{\omega_n\sqrt{1 - \zeta^2}} e^{-\zeta\omega_n t} \sin\left(\omega_n\sqrt{1 - \zeta^2}\,t\right) \tag{A.43}$$

$$g_{\text{step}}(t) = \frac{K}{\omega_n^2}\left\{1 - e^{-\zeta\omega_n t}\left[\cos\left(\omega_n\sqrt{1 - \zeta^2}\,t\right) + \frac{\zeta}{\sqrt{1 - \zeta^2}}\sin\left(\omega_n\sqrt{1 - \zeta^2}\,t\right)\right]\right\} \tag{A.44}$$

Figure A.11 shows the step response for a second-order system.

The system has a few properties that can be evaluated mathematically. For instance, if we take the derivative of the step response in (A.44) with respect to time and set it equal to zero, we can find the maximums and minimums of the system. The first one is at a time of

$$T_{\text{peak}} = \frac{\pi}{\omega_n\sqrt{1-\zeta^2}} \tag{A.45}$$

Another property that can be defined is the percentage overshoot. This is a measure of how much bigger the peak value is than the final value.

$$\text{Overshoot} = \frac{g_{\text{step}}(T_{\text{peak}}) - g_{\text{step}}(\infty)}{g_{\text{step}}(\infty)} = e^{\frac{-\zeta\pi}{\sqrt{1-\zeta^2}}} \tag{A.46}$$

We can also define the settling time as the time it takes the waveform to reach and stay within 2% of its final value. This means that the sinusoidal part of (A.44) must have an amplitude of 0.02. This can be approximated as

$$T_S \approx \frac{4}{\zeta\omega_n} \tag{A.47}$$

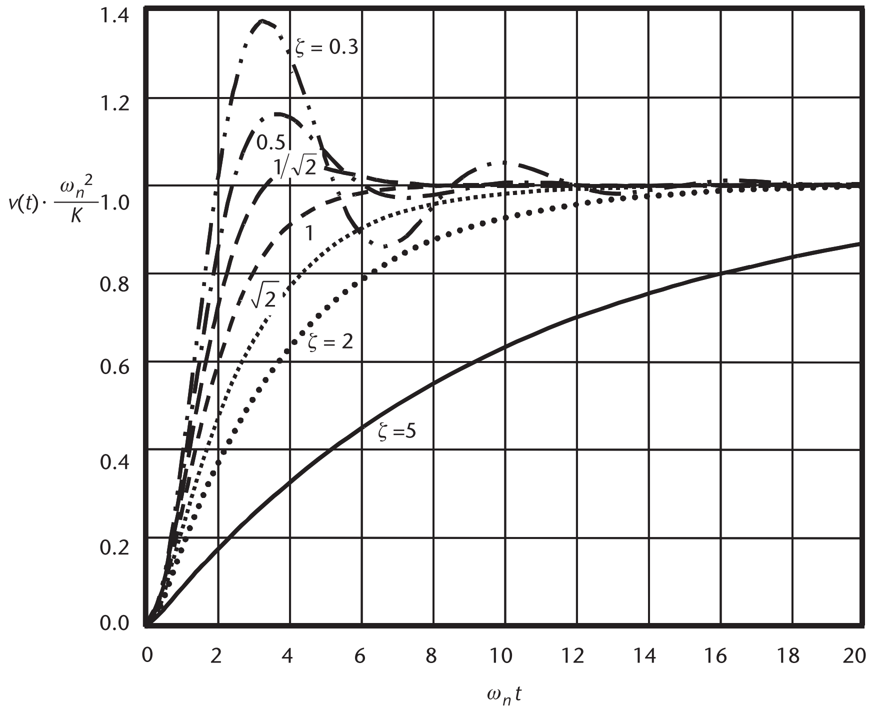


Figure A.11 The step response of a second-order system for different values of ζ.

Now we also need to determine what happens when we add zeros to the system. A general second-order transfer function that includes finite zeros would take the form

$$H(s) = \frac{K(s+\omega_{z1})\cdot(s+\omega_{z2})}{s^2+2\zeta\omega_n s+\omega_n^2} = G(s)(s+\omega_{z1})\cdot(s+\omega_{z2}) \tag{A.48}$$

$$H(s) = s^2G(s) + sG(s)(\omega_{z1}+\omega_{z2}) + G(s)\,\omega_{z1}\,\omega_{z2}$$

Here, $G(s)$ is the original transfer function given by (A.21). If there is only one zero, and it is at dc, then we have a pure differentiator, and the time domain equation will simply be the derivative of the original transfer function. The transient response of this system is simply the derivative of the transfer function without the zero. Otherwise, the transient response is a combination of both the derivative and the response of the system without the zero present, as well as a second derivative if two zeros are present. Note that zeros are not restricted to the left half-plane but can be in the right half-plane as well.

Thus, in the time domain, we have,

$$h(t) = \frac{d^2g(t)}{dt^2} + \frac{dg(t)}{dt}(\omega_{z1}+\omega_{z2}) + g(t)\,\omega_{z1}\,\omega_{z2} \tag{A.49}$$

In general, as more poles are added to a system, the system takes longer to settle. Zeros or differentiators can often be added to a system to speed up its response. This is especially useful in higher-order systems with multiple poles. Often, zeros can be added systems to them to make them more “second-order like” and, thus, have faster settling times or to improve the system stability.

Example A.3: Effect of a Zero

A first-order system with a zero is given by

$$F_{\text{zero}}(s) = \frac{K(s+\omega_z)}{(s+\omega_o)}$$

This system will have a step-transient response that is given by

$$f_{\text{step\_zero}}(t) = f_{\text{step}}(t)\,\omega_z + \frac{df_{\text{step}}(t)}{dt} = \frac{K\omega_z}{\omega_o} - \frac{K\omega_z}{\omega_o}e^{-\omega_o t} + Ke^{-\omega_o t} \tag{A.50}$$

Discrete systems can be handled as well. For example, a first-order discrete-time system that is described by (A.30) will have a response to a unit step given by

$$f_{\text{step}}(kT) = \frac{K}{1-e^{-\omega_o T}}u(kT) + \frac{K}{1-e^{\omega_o T}}e^{-\omega_o kT}u(kT) \tag{A.51}$$

This equation is obtained by taking the transfer function and multiplying by the z domain equation for a step function and then taking the inverse z transform to get the time domain output. Note that a transfer function in the frequency domain can also be transformed back into the time domain very simply by cross-multiplying, then replacing multiplications by z^0 with the subscript k and multiplications by z^{-1} by the subscript $k - 1$ (remember that z^{-1} is a unit delay). This is best illustrated with an example.

Example A.4: Transient Response of a Discrete-Time System
A continuous-time signal (a square wave) is given by

$$f_{\text{step}}(t) = u(t - T) - u(t - 3T) - [u(t - 5T) - u(t - 7T)]$$

This signal is applied to a system with a transfer function given by

$$\frac{v_{\text{out}}}{v_{\text{in}}} = \frac{1}{1 - z^{-1}}$$

Determine the time domain output of the system.
Solution: First, we take the transfer function and cross-multiply so that

$$v_{\text{out}} - v_{\text{out}} z^{-1} = v_{\text{in}}$$

Noting that z^{-1} is a unit delay means that, in the time domain, the previous equation can be rewritten as

$$v_{\text{out\_}k} = v_{\text{in\_}k} + v_{\text{out\_}k-1}$$

where k denotes the kth sampling instant. Figure A.12 shows a plot of the outputs noting that, at every sampling instant, the output is the sum of the last output plus the current input.

A.6 Feedback Systems

So far, we have discussed first- and second-order systems. Now we consider functional blocks connected in a feedback configuration instead of simply being connected in series. Feedback can be either positive or negative, although negative feedback is the more common and stable form of feedback. Figure A.13 shows a generic system that employs negative feedback. Here, a system with transfer function $A(s)$ (known as the forward gain) has its output fed back through another system with transfer function $B(s)$ (known as the feedback gain) whose output $v_{\text{feedback}}(s)$ is subtracted from the input signal (note that subtraction implies negative feedback). The signal fed into $A(s)$ is also known as the error signal $v_{\text{error}}(s)$. It is called an error signal because a negative feedback system operates in such a way that $v_{\text{in}}(s)$ and $v_{\text{feedback}}(s)$ are driven towards being equal, therefore minimizing the "error"

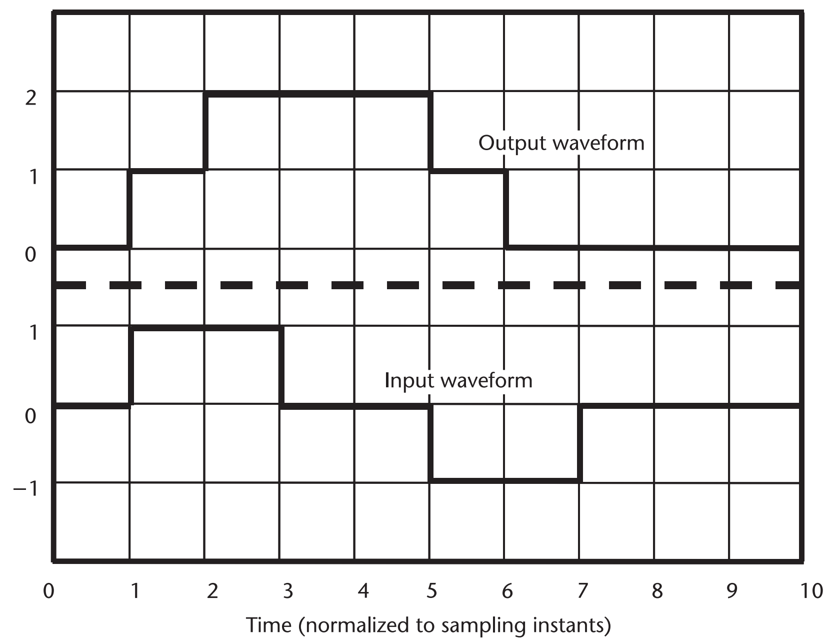


Figure A.12 Example of the transient response of a discrete system to a step input.

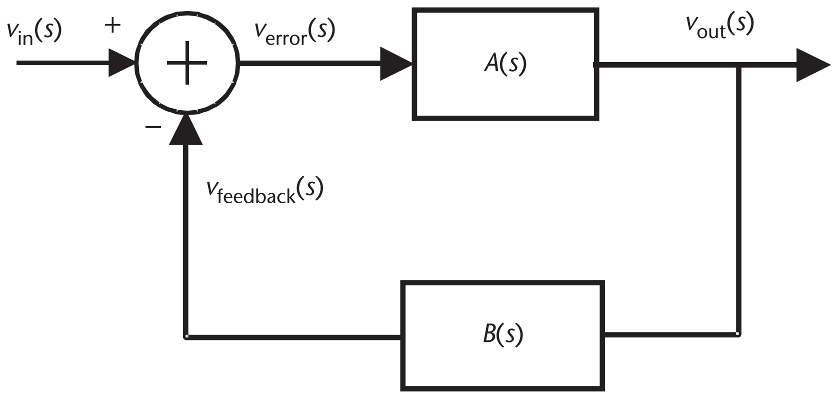


Figure A.13 Generic system with feedback.

signal. In the case of positive feedback, an error signal causes $v_{\text{in}}(s)$ and $v_{\text{feedback}}(s)$ to be driven further apart, towards the power-supply rails. Thus, one can see that getting the right number of inversions into the feedback path is very important as a simple change in sign can drastically change the outcome!

This system can be shown to have a transfer function from input to output given by

$$v_{\text{out}}(s) = A(s)v_{\text{error}}(s) = A(s)[v_{\text{in}}(s) - v_{\text{feedback}}(s)] = A(s)[v_{\text{in}}(s) - v_{\text{out}}(s)B(s)] \tag{A.52}$$

Solving for v_{out}/v_{in} gives

$$H(s) = \frac{v_{out}(s)}{v_{in}(s)} = \frac{A(s)}{1 + A(s)B(s)} \tag{A.53}$$

Thus, if the system is itself placed into a box, this is just another transfer function, and from the outside, it is impossible to know if there is feedback in it at all.

Note that, in general, we usually know the properties of $A(s)$ and $B(s)$ and are required to do math to solve for the overall closed-loop transfer function. This can become cumbersome for systems with high-order components, so a math program may need to be employed. Alternatively, the pole locations can be found, for example with a root-locus analysis. This is discussed in Section A.8.

Systems are also often a mixture of continuous and discrete parts, and a charge pump–based PLL is an example of such a system. Often, the inputs are the digital parts that control a continuous "real-world" system. For instance, the system shown in Figure A.14(a) can be represented by the model shown in Figure A.14(b). Note that if we assume that the sampling clocks are synchronized, the systems are equivalent. In this case, we can find the open-loop z transfer function of this system as

$$G(z) = (1 - z^{-1}) \cdot Z\left\{\frac{F(s)}{s}\right\} \tag{A.54}$$

where

$$Z\left\{\frac{F(s)}{s}\right\}$$

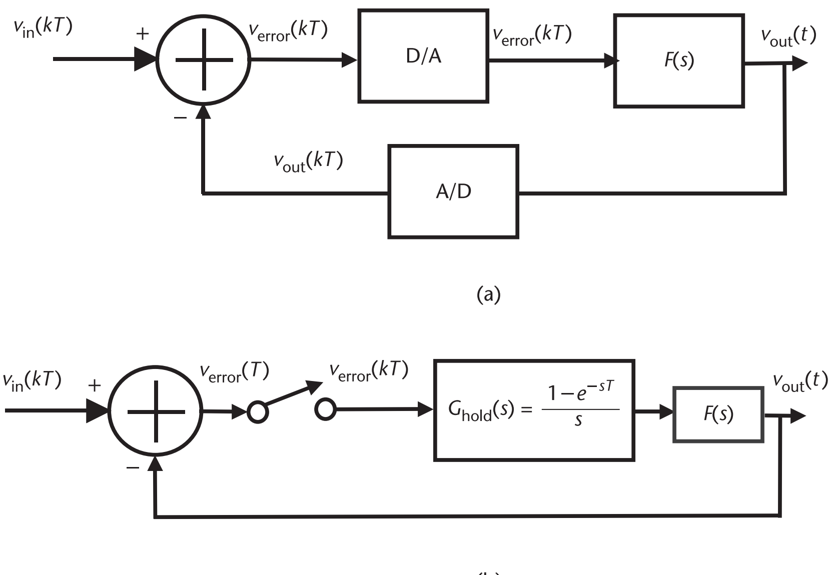


Figure A.14 (a) Mixed-signal feedback-control system, and (b) equivalent model.

is the z transform of the s domain function. Note that the term in front can be separated because, even though, in general, all multiplication must be done before the z transform is performed, multiplying by e^{-sT} (z^{-1}) is a special case where separation can be done before transformation to the z domain. This is, in fact, the Time-Shift Theorem (see Table A.1).

Therefore, the closed-loop transfer function for this system is

$$H(z) = \frac{G(z)}{1 + G(z)} \tag{A.55}$$

Figure A.15 shows examples of purely sampled systems in feedback, which, in this case, form discrete-time integrators. All circuits in Figure A.15 are equivalent, except for a different amount of delay added to the output. Thus, Figure A.15(a) has the minimal amount of delay, Figure A.15(b) is exactly equivalent to Figure A.15(c) and has an extra clock delay added to the output, and Figure A.15(d) has a half-delay added to the output. In the basic integrator, as in Figure A.15(a), the input is added to the previous output to form the new output. Thus, this is the sum of all inputs, which is exactly the function of an integrator. Let us see what this looks like from the z transform point of view. In this case,

$$v_o = v_x = v_i + v_o z^{-1} \tag{A.56}$$

This can be solved for v_o with the following result:

$$\frac{v_o}{v_i} = \frac{1}{1 - z^{-1}} \tag{A.57}$$

Similar results can be obtained for circuits with the extra full or half-delay as follows:

$$\text{Full delay: } \frac{v_o}{v_i} = \frac{z^{-1}}{1 - z^{-1}}, \text{ Half-delay: } \frac{v_o}{v_i} = \frac{z^{-1/2}}{1 - z^{-1}} \tag{A.58}$$

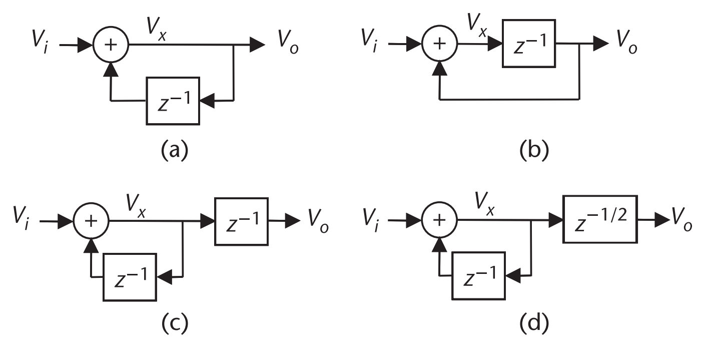


Figure A.15 Discrete feedback systems that implement an integrator: (a) no forward path delay, (b) one period of forward delay, (c) one period of forward delay outside feedback, and (d) half a period of delay.

A.7 Steady-State Error and the System Type

If there are any integrators in the feedback path [pure $1/s$ terms in either $A(s)$ or $B(s)$], this has an effect on the steady-state error of the system when a given input is applied. This is also a way to classify systems. If a system has no integrators [no poles in $A(s)$ or $B(s)$ at the origin], then the system is type zero. If it has one pole at the origin, it is type I; if it has two, then it is Type II; and so on.

The error signal of the feedback system is given by

$$v_{\text{error}}(s) = \frac{v_{\text{in}}(s)}{1 + A(s)B(s)} \tag{A.59}$$

For a type-zero system with a step input applied ($v_{\text{in}} = 1/s$), the steady-state error from the Final Value Theorem (see Table A.1) is given by

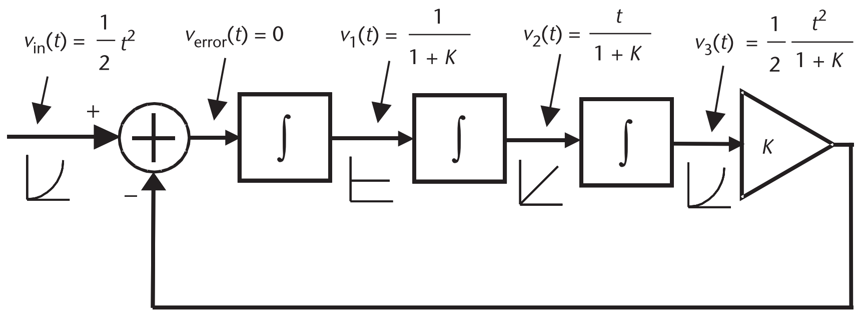
$$e_{\text{step}} = \lim_{s \to 0} s\left[\frac{1}{s} \cdot \frac{1}{1 + A(s)B(s)}\right] = \frac{1}{1 + A(0)B(0)} = \frac{1}{1 + K} \tag{A.60}$$

where K is the dc loop gain of the system. For a ramp input, as well as any parabolic input, the error is infinity.

If the system has an integrator in it, then the steady-state error is given by

$$e_{\text{step}} = \lim_{s \to 0} s\left[\frac{1}{s} \cdot \frac{1}{1 + \dfrac{A(s)B(s)}{s}}\right] = 0 \tag{A.61}$$

Another way to think about this is that an integrator can have a nonzero output with a zero input. Thus, if there is an input step, the output can adjust with a zero error signal. In general, to have a zero error signal, the system is required to have one more integrator than the order of the input transient. That is, a step is not a function of time, so it is order zero; a ramp is proportional to time, so it is order one; and a parabolic is proportional to the square of time, so it is order two. Thus, for a parabolic input, to have a zero error signal, a minimum of three integrators is required, as shown in Figure A.16. Note that, aside from the three integrators



$v_{\text{in}}(t) = \frac{1}{2}t^2$
$v_{\text{error}}(t) = 0$
$v_1(t) = \frac{1}{1+K}$
$v_2(t) = \frac{t}{1+K}$
$v_3(t) = \frac{1}{2}\frac{t^2}{1+K}$
+
−
∫
∫
∫
K

Figure A.16 How integrators influence error voltage.

in this figure, the rest of the loop transfer function is simply shown as a constant K. This is because, unless the transfer function has poles at the origin, only its dc gain will have any effect on the steady state of the system. You can see that in this figure, if one integrator is removed, v_1 will become the error signal, and if two are removed, v_2 will become the error signal, and so on. It should also be noted that Figure A.16 is primarily for illustration purposes and that, in practice, such a system could experience stability problems.

Note that the steady-state error for a sampled system can be found in exactly the same way using the z version of the final value theorem in the table.

A.8 Stability

So far, we have restricted our s domain discussion to systems with poles in the left half-plane. This is because these poles all have exponentials associated with them, where the exponentials decrease with time. If a pole were in the right half-plane, then it would have a growing exponential associated with it, and the system would be considered unstable. This is good for building oscillators, but most control systems should be designed to be stable. For the case where the system is sampled and poles are described in terms of z rather than s, the stability of these systems can be determined by pole location as well. Since

$$z = e^{Ts} \tag{A.62}$$

if we make the substitution that $s = \sigma + j\omega$, then

$$z = e^{T(\sigma + j\omega)} = e^{T\sigma}e^{j\omega T} = e^{T\sigma}[\cos(\omega T) + j\sin(\omega T)] = e^{T\sigma} \angle \omega T \tag{A.63}$$

We know that on the s plane, any pole with a real part ($\sigma > 0$) is unstable. From (A.63), $\sigma = 0$ on the s plane corresponds to a magnitude of one on the z plane. Therefore, for a sampled system to be stable, the poles must be within the unit circle on the z plane. Thus, the sampling rate can affect the stability of a system. For example, consider a feedback system that consists of a system with a first-order response. That is to say, assume that the system shown in Figure A.14 contains a system $F(s)$ given by (A.15). Now $G(z)$ is given by

$$G(z) = (1 - z^{-1}) \cdot Z\left[\frac{K}{s(s+\omega_o)}\right] = (1 - z^{-1}) \cdot \frac{K}{\omega_o} Z\left(\frac{1}{s} - \frac{1}{s+\omega_o}\right) \tag{A.64}$$

$$G(z) = (1 - z^{-1}) \cdot \frac{K}{\omega_o}\left(\frac{z}{z-1} - \frac{z}{z - e^{-\omega_o T}}\right) = \frac{K}{\omega_o}\left(\frac{1 - e^{-\omega_o T}}{z - e^{-\omega_o T}}\right)$$

Therefore, the closed-loop transfer function for this system is given by

$$H(z) = \frac{K}{\omega_o} \cdot \frac{1 - e^{-\omega_o T}}{z - \left[\left(1 + \frac{K}{\omega_o}\right)e^{-\omega_o T} - \frac{K}{\omega_o}\right]} \tag{A.65}$$

For this system to be stable, the system must have its pole inside the unit circle. Therefore,

$$\text{Pole} = \left(1 + \frac{K}{\omega_O}\right)e^{-\omega_o T} - \frac{K}{\omega_O} \tag{A.66}$$

For stability,

$$-1 < \left(1 + \frac{K}{\omega_O}\right)e^{-\omega_o T} - \frac{K}{\omega_O} < 1 \tag{A.67}$$

$$\frac{\dfrac{K}{\omega_O} - 1}{\dfrac{K}{\omega_O} + 1} < e^{-\omega_o T} < 1$$

For e^{-x} to be equal to one, x must equal zero. This can only happen if the sampling time goes to zero, or the cutoff frequency on the LPF is zero. So, for practical values ω_O and T, the right-hand condition is easily met. The other condition can be met if

$$-\frac{1}{\omega_O} \cdot \ln\left(\frac{\dfrac{K}{\omega_O} - 1}{\dfrac{K}{\omega_O} + 1}\right) > T \tag{A.68}$$

Thus, the system will be stable, provided the sampling period T is not too large. If it is too large (if the sampling frequency is too low), the system will become unstable.

For feedback systems, it is often easier to use the open-loop characteristics to determine stability. The classic definition of an oscillator (the Barkhausen criteria) states that (with reference to Figure A.13) for the system to be unstable, the loop gain must be equal to one at an angle of 0° or at a multiple of 360°. With an inversion in the feedback (negative feedback), the phase of $A(s)B(s)$ must be 180°. Thus, the system will be unstable, and oscillations will occur if the combination of $A(s)$ and $B(s)$ has a gain greater than unity when the phase shift is 180°. Thus, two typical stability criteria are either phase margin or gain margin.

Phase margin is defined as the amount of additional phase change needed at the unity-gain point of $A(s)B(s)$ to give a phase shift of 180°.

Gain margin is defined as the amount of additional gain at the frequency where $A(s)B(s)$ has a phase shift of 180° to start an oscillation.

A.9 Root Locus

Often with a feedback system, we know the transfer functions for $A(s)$ and $B(s)$, but algebra must be performed to find the closed-loop transfer function of the whole system. If $A(s)$ and $B(s)$ are of high order, finding the poles of the closed-loop transfer function can be tedious. Thus, if a computer is not handy, it is often useful to have a technique for finding the poles quickly with pen and paper. Often, we would like to know where the poles of the system are as a function of the dc loop gain K. If K is assumed to be in the feedforward path, then the closed-loop transfer function is

$$T(s) = \frac{KA(s)}{1 + KA(s)B(s)} \tag{A.69}$$

The poles of this system can be found from

$$KA(s)B(s) = -1 \tag{A.70}$$

or, more explicitly, when

$$\angle A(s)\angle B(s) = (2k + 1) \cdot 180° \tag{A.71}$$

where k is a positive integer and when

$$K = \frac{1}{|A(s)||B(s)|} \tag{A.72}$$

Therefore, by substituting values for s into $A(s)$ and $B(s)$ and making use of (A.70) and (A.71), we can test to see if any point happens to be a pole of the closed-loop system for some value of K. This is still quite tedious, so we need some more information to help us determine how many points we need to test to get a good idea of the pole locations.

We can rewrite (A.69) with $A(s)$ and $B(s)$ replaced explicitly with their denominator (D) and numerator (N) components:

$$T(s) = \frac{KN_A D_B}{D_A D_B + KN_A N_B} \tag{A.73}$$

Now, if we take the limit as K approaches zero, then

$$T(s)\Big|_{K \to 0} = \frac{KN_A D_B}{D_A D_B} \tag{A.74}$$

Therefore, for very small values of K, the closed-loop poles are equal to the open-loop poles.

Next, we will see if poles lie on the real axis. First, note that the poles and zeros for any $A(s)$ or $B(s)$ of interest to us will either lie on the real axis or else they will be conjugate pairs. Thus, we can expect that the plot of the open-loop (as well as the closed-loop) poles and zeros will be symmetric about the real axis.

Now, we test whether any point on the real axis at $s = a$ is a closed-loop pole. If it is, the first condition is that, for that value of s, the phase shift must add up to some multiple of 180°. Since the point is on the real axis, any conjugate pair of poles or zeros will add no net phase shift (the two phases will be equal but opposite in sign). Now if any pole or zero is sitting on the axis to the left when $s = a$ is substituted in, it will have a positive real value and will not contribute any phase shift. Any pole or zero located to the right of the point, however, will have a negative real value and will contribute a phase shift of 180°. Therefore, any point of the real axis that is to the right of an odd number of poles and zeros will be a pole of the system for some value of K.

Thus, from the preceding, we know where the closed-loop poles can be found when the gain is low, but where do they end up for high gain? To determine this, we take the limit as K approaches infinity:

$$T(s)\Big|_{K\to\infty} = \frac{N_A D_B}{N_A N_B} \tag{A.75}$$

Thus, for very high values of loop gain, the closed-loop poles of the system end up being equal to the open-loop zeros of the system. However, what about the case where the open-loop system has no zeros? Actually, the system always has zeros, but if they are not explicitly present, then they end up at infinity. So, where exactly do the poles go if they head towards infinity? The answer is that they head to infinity at an angle of [3]

$$\theta = \frac{(2k+1)\pi}{\#\text{ finite poles} - \#\text{ finite zeros}} \tag{A.76}$$

and these asymptotes intercept the real axis at a point

$$\sigma = \frac{\sum \text{finite poles} - \sum \text{finite zeros}}{\#\text{ finite poles} - \#\text{ finite zeros}} \tag{A.77}$$

Example A.5: Plotting the Root Locus

Plot the root locus for the variable K in the following open-loop-gain transfer function:

$$F(s) = \frac{K \cdot s \cdot (s+4) \cdot (s+5)}{(s+1)(s+6)(s+7)(s+1+j)(s+1-j)}$$

Solution: The root locus for this system begins with the open-loop poles and zeros. These can be found by inspection from this conveniently factored expression. The poles are located at −1, −6, −7, and $-1 \pm j$. These are plotted in Figure A.17 using Xs. Also, there are three finite open-loop zeros at 0, −4, and −5. These are plotted in the figure using Os. This also means that the system has two zeros at infinity, which in turn means that there will be two root-locus lines that head towards infinity. The next thing to determine is where the root locus exists on the real axis. Remembering that it exists to the right of an odd number of real axis poles and zeros, we can determine that it will exists on the real axis between 0 and −1, between −4 and −5, and between −7 and −6. This makes the plot of the

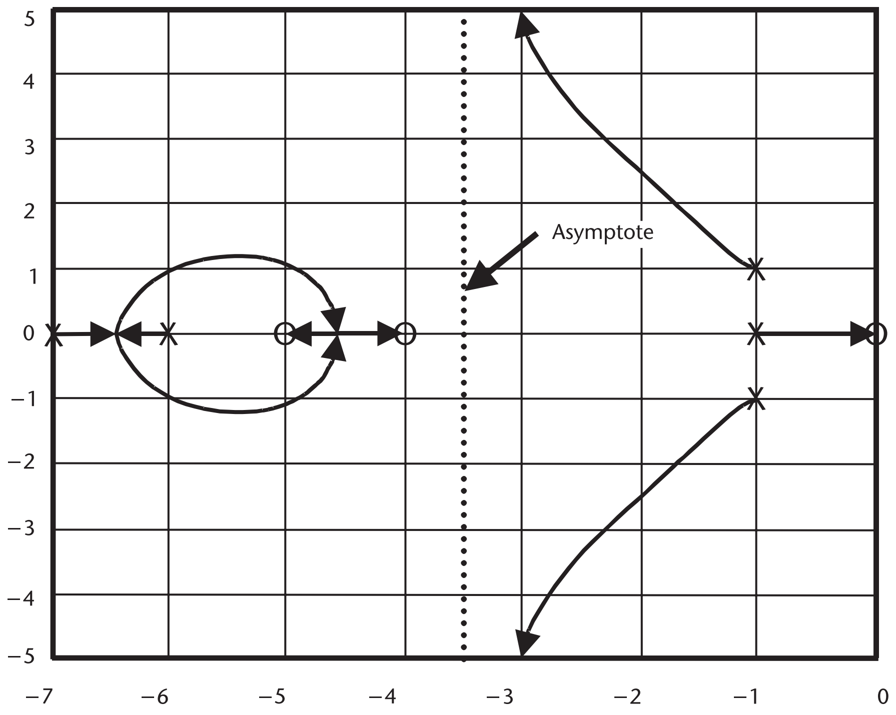


Figure A.17 Example of a root-locus plot.

root-locus line that begins at –1 fairly obvious. It will run from the pole at –1 to the zero at 0. Next, we need to find the real axis intercept of the asymptotes of the two lines that will go to infinity. From (A.77) it will be at

$$\sigma = \frac{[(-7) + (-6) + (-1) + (-1 + j) + (-1 - j)] - [(-5) + (-4) + 0]}{5 - 3} = -3.5$$

From (A.76), it is easy to see that these will go in the direction of 90° and 270°. These will be the root-locus lines for the poles starting at $-1 \pm j$. Now there are only two poles left for which to draw root-locus lines. Those are the ones at –6 and –7. These lines will both exist on the real axis between –6 and –7 until they meet somewhere in the middle. Once they meet, they will break away from the real axis and travel in the complex plane until they rejoin between –4 and –5. Note that the root locus has to be symmetric about the real axis, the two poles have to end up at –4 and –5, and they have to be on the real axis between –4 and –5, so this is the only choice. Once they break into the real axis again, they both split, and one heads to –4 and one heads to –5. The complete root-locus plot is shown in Figure A.17.

References

[1] Boyce, W. E., and R. C. DiPrima, *Elementary Differential Equations and Boundary Value Problems*, 5th ed., New York: John Wiley & Sons, 1992.

[2] Ogata, K., *System Dynamics*, 2nd ed., Upper Saddle River, NJ: Prentice Hall, 1992.

[3] Nise, N. S., *Control Systems Engineering*, 2nd ed., Redwood City, CA: The Benjamin/Cummings Publishing Co., 1995.

[4] Sklar, B., *Digital Communications: Fundamentals and Applications*, 2nd ed., Upper Saddle River, NJ: Prentice Hall, 2001.

[5] Gregorian, R., and G. C. Temes, *Analog MOS Integrated Circuits for Signal Processing*, New York: John Wiley & Sons, 1986.

APPENDIX B

A Review of Transistor Models

B.1 Introduction

This appendix reviews very briefly some basic transistor theory. The authors assume that anyone reading it will have a basic working knowledge of circuit theory from an undergraduate electrical engineering program or the equivalent. A short appendix cannot hope to replace the many detailed texts available on this subject, but it is included here as a reference for a few very common equations and basic results.

B.2 The Basics of CMOS Transistors

Traditionally, bipolar transistors are preferred for high-speed circuits due to the higher values of transconductance achievable for a given amount of bias current. However, it is often necessary to use a process that can also be used to implement back-end digital or DSP functions. In such cases, a BiCMOS or straight CMOS process is preferred. If a BiCMOS process is used, bipolar transistors can be used for RF, possibly adding PMOS transistors for power-control functions. However, for economic reasons, or due to the need to use a particular CMOS-only process to satisfy the back-end requirements, it may be necessary to implement all circuits in pure CMOS. Figure B.1 shows basic PMOS and NMOS transistors. An NMOS transistor is made by growing an oxide layer on top of a p type substrate [1]. The oxide layer is patterned from the gate (G) of the transistor. n+ regions are then implanted on either side of the gate to form the source (S) and drain (D) of the transistor. The gate also has a conductive layer (typically polysilicon) placed on top of the oxide to form an electrical connection. When electrical connections are made to each of these three regions (as well as to the substrate), this structure forms a transistor. In operation, as discussed in more detail in the next section, the voltage on the gate is used to form a channel under the gate and to control the current (or, equivalently, the resistance) between the source and drain regions. The PMOS transistors are similar except they must be made in an n type substrate (which will likely be an n well), and they have p+ implants rather than n+ implants.

B.2.1 Basic DC Biasing Characteristics

The drain characteristic curves for a CMOS transistor are shown in Figure B.2. When a positive voltage is applied to the gate of the NMOS device, electrons are

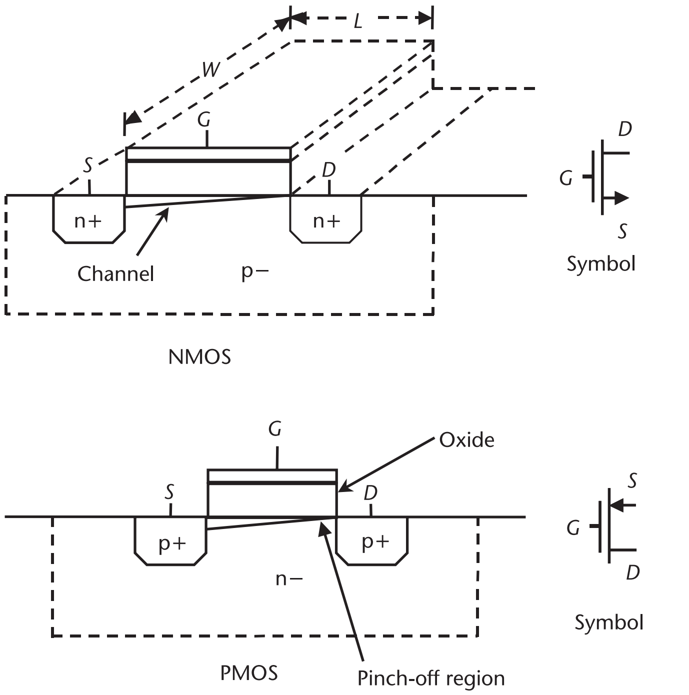


Figure B.1 Basic NMOS and PMOS transistor structures.

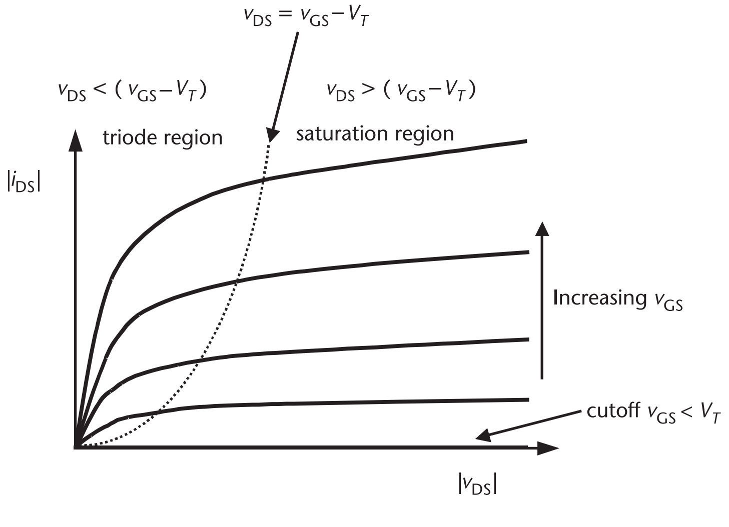


Figure B.2 Basic transistor voltage and current characteristics.

attracted towards the gate. With sufficient voltage, an n channel is formed under the oxide, allowing electrons to flow between the drain and the source under the control of the gate-source voltage v_{GS}. Thus, as gate voltage is increased relative to the source voltage, current increases. For small applied v_{DS}, with constant v_{GS}, the current between drain and source is related to the applied v_{DS}. For very low v_{DS}, the relationship is nearly linear; thus, the transistor behaves much like a resistor. For sufficiently large v_{DS}, the channel becomes restricted at the drain end (pinched off), as shown in Figure B.1. For larger v_{DS}, current is saturated and remains nearly independent of v_{DS}. This means the output resistance is very high, and the transistor acts like a constant current source.

The operation of PMOS is similar to that of NMOS except that negative v_{GS} is applied. This attracts holes to form a conducting p channel. The characteristic curves for PMOS and NMOS are similar if the absolute value is taken for current and voltage.

B.2.2 Basic CMOS Square Law Equations

We now show some simplistic equations for calculating model parameters and doing simple hand calculations. In the saturation region of operation, the drain-source current can be described by the simple square law equation:

$$i_{\mathrm{DS}} = \frac{\mu C_{\mathrm{ox}}}{2}\left(\frac{W}{L}\right)(v_{\mathrm{GS}} - V_T)^2(1 + \lambda v_{\mathrm{DS}}) \tag{B.1}$$

where V_T is the threshold voltage, μ is the electron mobility in silicon (or hole mobility in the case of PMOS), C_{ox} is the gate capacitance per unit area, and λ is the output slope factor given by

$$\lambda = \frac{K}{2L\sqrt{V_{\mathrm{DS}} - (V_{\mathrm{GS}} - V_T) + \Phi_o}} \tag{B.2}$$

where Φ_o is the built-in open-circuit voltage of silicon [2] and K is a constant dependent on the doping levels. λ can be related to the drain-source conductance of the device, which is given by

$$g_{\mathrm{DS}} = \frac{di_{\mathrm{DS}}}{dv_{\mathrm{DS}}} = I\lambda \tag{B.3}$$

and, therefore, is proportional to the length of the device.

Various attempts have been made to include other effects, such as mobility degradation and velocity saturation; for example [3],

$$i_{\mathrm{DS}} = \frac{\mu C_{\mathrm{ox}}}{2}\left(\frac{W}{L}\right)\frac{(v_{\mathrm{GS}} - V_T)^2}{1 + \alpha(v_{\mathrm{GS}} - V_T)^2}(1 + \lambda v_{\mathrm{DS}}) \tag{B.4}$$

where α approximately models the combined mobility degradation and velocity saturation effects given by [4]

$$\alpha = \theta + \frac{\mu_0}{2nv_{\mathrm{sat}}L} \tag{B.5}$$

where θ is the mobility-reduction coefficient, and v_{sat} is the saturation velocity. We note that for small values of α or small overdrive voltage $(v_{\text{GS}} - V_T)$, (B.4) becomes the familiar square law equation as given by (B.1).

The transconductance is given by the derivative of the current with respect to the gate-source voltage. For the simple square law equation, this becomes

$$g_m = \frac{di_{\text{DS}}}{dv_{\text{GS}}} \mu C_{\text{ox}}\left(\frac{W}{L}\right)(v_{\text{GS}} - V_T)(1 + \lambda v_{\text{DS}}) \tag{B.6}$$

This can also be shown to be equal to (note that the λ term has been left out)

$$g_m = \sqrt{2\mu C_{\text{ox}}\frac{W}{L} I_{\text{DS}}} \tag{B.7}$$

In the triode region of operation, current is given by

$$i_{\text{DS}} = \mu C_{\text{ox}}\left(\frac{W}{L}\right)\left(v_{\text{GS}} - V_T - \frac{v_{\text{DS}}}{2}\right)v_{\text{DS}}(1 + \lambda v_{\text{DS}}) \tag{B.8}$$

In practice, for high-speed design, short-channel devices are used. The equations for these devices are poor; thus, it is necessary to use simulators to find the curves, transconductances, impedances, and so forth, or to use more complicated models, such as those presented in [5].

B.2.3 The Body Effect

If the source-substrate voltage of the transistor is not zero, then the V_T of the device or, equivalently, the effective transconductance of the device, is changed. This is called the body effect. It is often modeled as an additional transconductance g_s in parallel with g_m. Provided the transistors can be placed in their own wells, the body effect can be avoided by shorting the substrate and source together. However, for transistors in separate wells, this can reduce matching between transistors.

B.2.4 High-Frequency Effects

Any transistor will have capacitance associated with it. The capacitance of most concern in a MOSFET is the gate-source capacitance. It has a value that can be approximated by

$$C_{\text{gs}} = \gamma W L C_{\text{ox}} \tag{B.9}$$

where γ is a parameter that corresponds to the fraction of the gate not in pinch off. In a long-channel device, γ will have a value of one at $V_{\text{DS}} = 0$ and decrease to approximately 2/3 in saturation. There are many other capacitors in the transistor as well, including gate-drain capacitance, drain-substrate capacitance, and source-substrate capacitance, but these are less easily calculated from simple formulas and are better handled with simulators.

There is a typical figure of merit called the transit frequency, or f_T, used to describe the speed of a transistor. f_T is the frequency at which the short circuit current gain (ratio of drain to gate current) is equal to one and is given by

$$f_T \approx \frac{g_m}{2\pi C_{\mathrm{gs}}} = \frac{\mu C_{\mathrm{ox}}\left(\frac{W}{L}\right)(v_{\mathrm{GS}} - v_T)}{2\pi\gamma WLC_{\mathrm{ox}}} = \frac{\mu(v_{\mathrm{GS}} - v_T)}{2\pi\gamma L^2} \tag{B.10}$$

Note that in many processes, f_T is nearly independent of width for the same current density in the drain (but f_T is always a strong function of current).

B.2.5 Thermal Noise

A resistor is one of the most common noise sources in a circuit. Noise in resistors is generated by thermal energy causing random electron motion [6–8]. The thermal noise in a resistor is given by

$$v_n^2 = 4kTR \tag{B.11}$$

where T is the temperature in kelvins of the resistor, k is Boltzmann's constant (1.38×10^{-23} j/K), and R is the value of the resistor. Thermal noise is white noise, meaning it has a constant power spectral density with respect to frequency. The model for noise in a resistor is shown in Figure B.3.

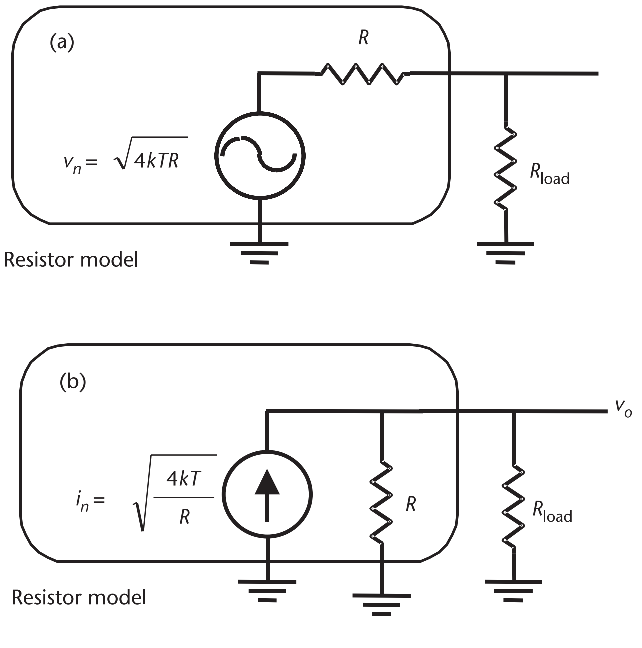


Figure B.3 Resistor noise model: (a) with a voltage source, and (b) with a current source.

In a MOSFET, the channel under the gate acts like a resistor, so, just like a resistor, it generates thermal noise. The drain noise current produced by a MOSFET is given by

$$i_{\mathrm{dn}}^2 = 4kT\gamma g_m \tag{B.12}$$

B.2.6 Shot Noise

Shot noise is due to the discrete nature of charge carriers as they pass a potential barrier, for example a pn junction. Shot noise is described by

$$i_{\mathrm{n\_shot}} = \sqrt{2qI} \tag{B.13}$$

where I is the current flow through the pn junction. The frequency spectrum of shot noise is also white.

B.2.7 1/*f* Noise

This type of noise is also called flicker noise, or excess noise. 1/*f* noise is due to variation in conduction mechanisms, for example, fluctuations of surface effects, such as the filling and emptying of traps and of recombination and generation mechanisms. Typically, in a MOSFET, 1/*f* noise can be modeled as a noise voltage on the gate; the PSD of 1/*f* noise is inversely proportional to frequency and is given by the following equation:

$$\overline{v_{\mathrm{nf}}^2} = \frac{K_f}{WLC_{\mathrm{ox}} f^{\alpha}} \tag{B.14}$$

where K_f is a process constant and α is approximately 1. As demonstrated by (B.14), 1/*f* noise in a MOSFET is inversely proportional to its area. It should also be noted that PMOS transistors typically have lower 1/*f* noise than NMOS transistors.

B.2.8 Gate Noise

The channel noise will also couple in the gate, appearing as current fluctuations in the gate. If both channel noise and gate noise are modeled in a transistor, then they will, to some degree, be correlated. This gate noise current is given by [9]

$$\overline{i_{\mathrm{ng}}^2} = \frac{4kT\delta\omega^2 C_{\mathrm{gs}}^2}{5g_m} \tag{B.15}$$

where δ is dependent on device geometry but, in long channel devices, will have a value of about twice γ, or 4/3.

Another source of noise in the gate is thermal noise due to gate resistance. This gate resistance can be calculated from the dimensions of the gate and the gate resistivity ρ by

$$r_g = \frac{1}{3}\rho\frac{W}{L} \tag{B.16}$$

for a gate with a contact on one side. Here, ρ is the effective resistivity of the gate poly with typical values between 10 and 20 $\mu\Omega$cm. We note that the gate poly by itself would have a resistance of $\rho W/Lt$, where t is the effective thickness of the silicided poly gate, with a typical value of 0.1 μm. The factor of 1/3 in (B.16) comes from the fact that the transistor current is flowing under all regions of the gate. The series resistance varies from 0Ω near the contact to $\rho W/Lt$ for the far end of the gate, with an effective value given by (B.16). If the gate is contacted on both sides, the effective resistance drops by a further factor of four such that

$$r_g = \frac{1}{12}\rho\frac{W}{L} \tag{B.17}$$

B.2.9 CMOS Small-Signal Model, Including Noise

The equations and discussion so far can be used to construct a model for the small-signal behavior of a CMOS transistor. Figure B.4 shows a model that includes all the characteristics discussed so far. Note that substrate capacitance is included in this model and that the body effect is modeled with transconductance g_s.

B.3 Bipolar Transistors

Figure B.5 shows a cross section of a basic npn bipolar transistor. The collector is formed by epitaxial growth in a p– substrate (the n– region). A p region inside the collector region forms the base region, then an n+ emitter region is formed inside the base region. The basic transistor action all takes place directly under the emitter. This can be called the intrinsic transistor. The intrinsic transistor is connected through the diffusion regions to the external contacts. More details on advanced bipolar structures, for example, using SiGe HBTs and double-poly self-aligned processes, can be found in the literature [1, 10].

When the transistor is being used as an amplifying device, the base-emitter junction is forward biased while the collector-base junction is reverse biased,

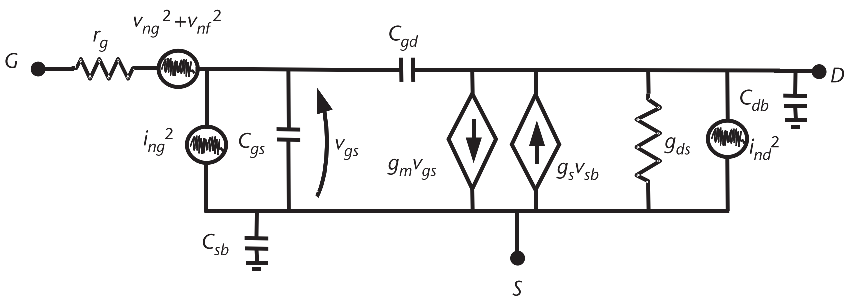


Figure B.4 A small-signal model for a CMOS transistor including noise.

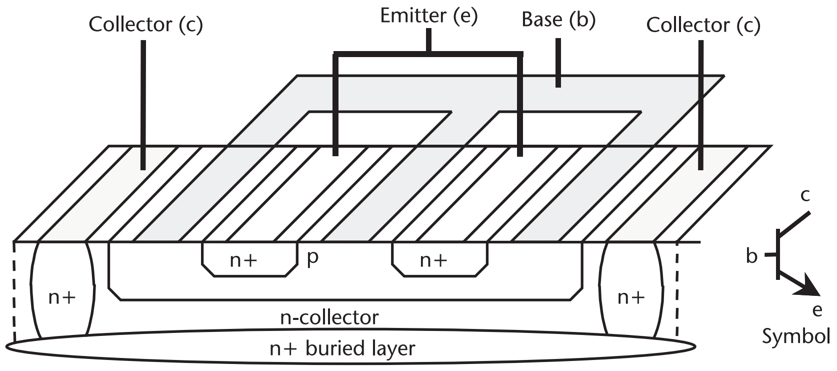


Figure B.5 Structure of a bipolar transistor.

meaning the collector is at a higher voltage than the base. This bias regime is known as the forward active region. Electrons are injected from the emitter into the base region. Because the base region is narrow, most electrons are swept into the collector rather than going to the base contact. This is equivalent to conventional (positive) current from collector to emitter. Some holes are back-injected into the emitter, and some electrons recombine in the base, resulting in a small base current that is directly proportional to collector current $i_c = \beta i_b$. Thus, the overall concept is that collector current is controlled by a small base current. The collector current can also be related to the base-emitter voltage in this region of operation by

$$I_C = I_S e^{\frac{V_{\text{BE}}}{v_T}} \tag{B.18}$$

where I_S is a constant known as the saturation current, V_{BE} is the dc bias between the base and emitter, and v_T is the thermal voltage given by

$$v_T = \frac{kT}{q} \tag{B.19}$$

where q is the electron charge, T is the temperature in kelvins, and k is Boltzmann's constant. The thermal voltage is approximately equal to 25 mV at a temperature of 290K, close to room temperature.

Figure B.6 shows the collector characteristics for a typical bipolar transistor. The transistor has two other regions of operation. When the base-emitter junction is not forward biased, the transistor is cut off. The transistor is in the saturated region if both the base-emitter and collector-emitter junction are forward biased. In saturation, the base is flooded with minority carriers. This generally leads to a delayed response when the bias conditions change to another region of operation. In saturation, V_{CE} is typically less than a few tenths of a volt. Note that in the active region, the collector current is not constant. There is a slope to the current-versus-voltage curve, indicating that the collector current will increase with collector-emitter voltage. The slopes of all the lines are such that they will meet at a negative voltage V_A called the Early voltage. This voltage can be used to characterize the transistor output impedance given by

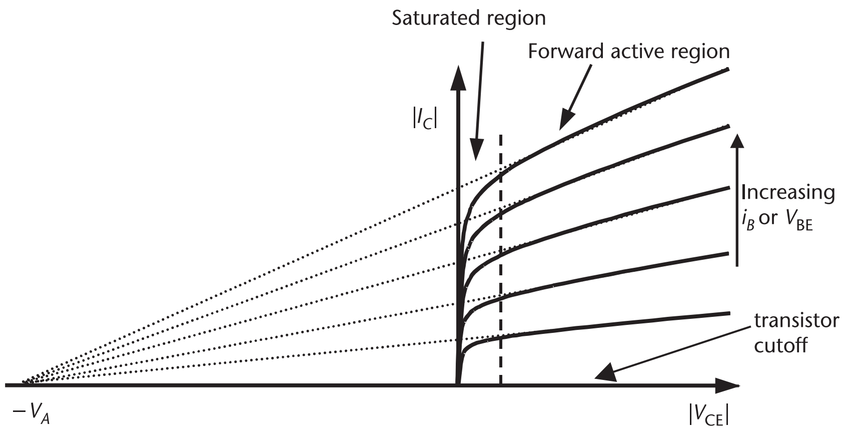


Figure B.6 Bipolar transistor curves.

$$r_o = \frac{V_A}{I_C} \tag{B.20}$$

The short circuit current gain β is given by

$$\beta = \underbrace{\frac{i_c}{i_b}}_{\text{small-signal}} = \underbrace{\frac{\Delta I_C}{\Delta I_B}}_{\Delta\text{large-signal}} \tag{B.21}$$

noting that currents can be related by

$$i_c + i_b = i_e \tag{B.22}$$

Transconductance g_m is given by

$$g_m = \frac{i_c}{v_\pi} = \frac{I_C}{v_T} = \frac{I_c q}{kT} \tag{B.23}$$

where I_C is the dc collector current and v_π is the base-emitter voltage.

At low frequency, where the transistor input impedance is resistive, v_π and i_b can be related by

$$\frac{v_\pi}{i_b} = r_\pi = \frac{\beta}{g_m} \tag{B.24}$$

(neglecting current due to finite output resistance).

In a bipolar transistor, there are two major capacitors of concern. The first (larger capacitor) is the base-emitter capacitance called C_π, and the second is the base-collector capacitance called C_μ. Both of these are junction capacitances, and their values depend on device geometries and the technology in question. The f_T of a bipolar transistor is given by

$$f_T = \frac{g_m}{2\pi(C_\pi + C_\mu)} \approx \frac{g_m}{2\pi C_\pi} = \frac{I_C}{2\pi C_\pi V_T} \tag{B.25}$$

There are four major sources of noise in a bipolar transistor: base shot noise and collector shot noise (due to the two pn junctions), base resistance, and 1/f noise. In a bipolar transistor, 1/f noise is best modeled as a current injected at the base and is given by the following equation:

$$\overline{i_{\mathrm{bf}}^2} = KI_C^m \frac{1}{f^\alpha} \tag{B.26}$$

where m is between 0.5 and 2, α is about equal to 1, and K is a process constant. Typically 1/f noise is several orders of magnitude less important for bipolar transistors compared to MOS transistors.

With the addition of these noise sources, a small-signal model for a bipolar transistor can be constructed and is shown in Figure B.7. Some of the major results of this appendix are summarized for convenience in Table B.1.

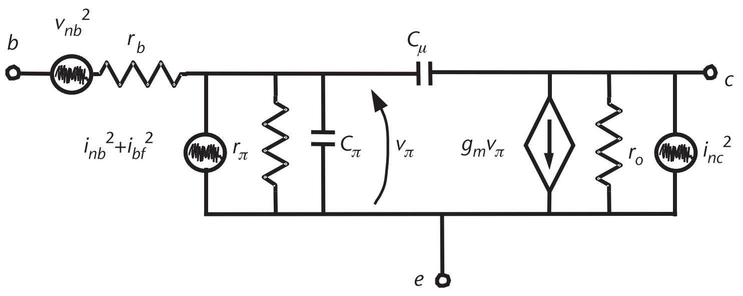


Figure B.7 Bipolar transistor model.

Table B.1 Summary of Important Transistor Characteristics

Description	*CMOS Formula*	*Bipolar Formula*
Collector/drain current (in the active/saturation region)	$i_{\text{DS}} = \frac{\mu C_{\text{ox}}}{2}\left(\frac{W}{L}\right)\frac{(v_{\text{GS}} - V_T)^2}{1 + \alpha(v_{\text{GS}} - V_T)^2} \times (1 + \lambda v_{\text{DS}})$	$I_C = I_S e^{\frac{v_{\text{BE}}}{v_T}}$
Transconductance	$g_m = \sqrt{2\mu C_{\text{ox}}\frac{W}{L}I_{\text{DS}}}$ $= \mu C_{\text{ox}}\left(\frac{W}{L}\right)(v_{\text{GS}} - v_T)(1 + \lambda v_{\text{DS}})$	$g_m = \frac{I_C}{v_T} = \frac{I_c q}{kT}$
f_T	$f_T \approx \frac{g_m}{2\pi C_{\text{gs}}} = \frac{\mu(v_{\text{GS}} - v_T)}{2\pi\gamma L^2}$	$f_T \approx \frac{g_m}{2\pi C_\pi}$
Collector shot noise/ drain noise	$i_{\text{dn}}^2 = 4kT\gamma g_m$	$i_{\text{nc}}^2 = 2qI_c$
Base shot noise/ gate noise current	$\overline{i_{\text{ng}}^2} = \frac{4kT\delta\omega^2 C_{\text{gs}}^2}{5g_m}$	$i_{\text{nb}}^2 = 2qI_b$
Base resistance	N/A	$r_\pi = \frac{\beta}{g_m}$
Base resistance noise/ gate resistance noise	$v_{\text{ng}}^2 = 4kTr_g$	$v_{\text{nb}}^2 = 4kTr_b$
Collector emitter/ drain-source resistance	$g_{\text{DS}} = I_{\text{ds}}\lambda = \frac{KI_{\text{ds}}}{2L\sqrt{V_{\text{DS}} - (V_{\text{GS}} - V_T) + \Phi_o}}$	$r_o = \frac{V_A}{I_C}$
Base emitter/ gate-source capacitance	$C_{\text{gs}} = \gamma WLC_{\text{ox}}$	C_π (no easy formula)
$1/f$ noise sources	$\overline{v_{\text{nf}}^2} = \frac{K}{WLC_{\text{ox}} f^\alpha}$	$\overline{i_{\text{bf}}^2} = KI_C^m \frac{1}{f^\alpha}$

References

[1] Taur, Y., and T. H. Ning, *Fundamentals of Modern VLSI Devices*, Cambridge, U.K.: Cambridge University Press, 1998.

[2] Sze, S. M., *High Speed Semiconductor Devices*, New York: John Wiley & Sons, 1990.

[3] Sedra, A. S., and K. C. Smith, *Microelectronic Circuits*, 4th ed., New York: Oxford University Press, 1998.

[4] Terrovitis, M. T., and R. G. Meyer, "Intermodulation Distortion in Current-Commutating CMOS Mixers," *IEEE J. of Solid-State Circuits*, Vol. 35, No. 10, October 2000, pp. 1461–1473.

[5] Sakurai, T., and R. Newton, "Alpha-Power Law MOSFET Model and Its Applications to CMOS Inverter Delay and Other Formulas," *IEEE J. of Solid-State Circuits*, Vol. 25, No. 2, April 1990, pp. 584–594.

[6] Papoulis, A., *Probability, Random Variables, and Stochastic Processes*, New York: McGraw-Hill, 1984.

[7] Sze, S. M., *Physics of Semiconductor Devices*, 2nd ed., New York: John Wiley & Sons, 1981.

[8] Gray, P. R., et al., *Analysis and Design of Analog Integrated Circuits*, 4th ed., New York: John Wiley & Sons, 2001.

[9] van der Ziel, A., *Noise in Solid-State Devices and Circuits*, New York: John Wiley & Sons, 1986.

[10] Plummer, J. D., P. B. Griffin, and M. D. Deal, *Silicon VLSI Technology: Fundamentals, Practice, and Modeling*, Upper Saddle River, NJ: Prentice Hall, 2000.

About the Authors

John Rogers received a B.Eng. in 1997, an M.Eng. in 1999, and a Ph.D. in 2002, all in electrical engineering from Carleton University, Ottawa, Canada. During his master's degree research, he was a resident researcher at Nortel Networks' Advanced Technology Access and Applications Group, where he did exploratory work on voltage-controlled oscillators (VCOs). From 2000 to 2002, he collaborated with SiGe Semiconductor Ltd. while pursuing his Ph.D. on low voltage Radio Frequency Integrated Circuits (RFICs). Concurrent with his Ph.D. research, he worked as part of a design team that developed a cable modem integrated circuit (IC) for the DOCSIS standard. From 2002 to 2004, he collaborated with Cognio Canada Ltd. doing research on MIMO RFICs for wireless local area network (WLAN) applications. He is currently an assistant professor at Carleton University. He is the coauthor of *Radio Frequency Integrated Circuit Design* (Artech House, 2003), and his research interests are in the areas of RFIC and mixed-signal design for wireless and broadband applications. Dr. Rogers received an IBM faculty partnership award in 2004, an Institute of Electrical and Electronics Engineers (IEEE) Solid-State Circuits Predoctoral Fellowship in 2002, and the BCTM best student paper award in 1999. He holds five U.S. patents and is a member of the Professional Engineers of Ontario and the IEEE.

Calvin Plett received a B.A.Sc. in electrical engineering from the University of Waterloo, Canada, in 1982, and an M.Eng., and a Ph.D. from Carleton University, Ottawa, Canada, in 1986 and 1991, respectively. Prior to 1982, he worked for a number of companies, including spending nearly four years with Atomic Energy of Canada and shorter periods with Xerox, Valcom, Central Dynamics, and Philips. From 1982 to 1984, he worked with Bell-Northern Research doing analog circuit design. In 1989, he joined the Department of Electronics, Carleton University, Ottawa, Canada, where he is now an associate professor. For some years, he did consulting work for Nortel Networks in RFIC design. For the last number of years, he has been involved in collaborative research, which has involved numerous graduate and undergraduate students and various companies, including Nortel Networks, SiGe Semiconductor, Philsar, Conexant, Skyworks, IBM, and Gennum. He has authored or coauthored more than 60 technical papers, which have appeared in international journals and conferences. He is a coauthor of *Radio Frequency Integrated Circuit Design* (Artech House, 2003). His research interests include the design of analog and RFICs, including filter design, and communications applications. Dr. Plett is a member of AES and the PEO and a senior member of the IEEE. He has been the faculty advisor to the student branch of the IEEE at Carleton

University for about 14 years. He coauthored papers that won the best student paper awards at BCTM 1999 and at RFIC 2002.

Foster Dai received a B.S. in physics from the University of Electronic Science and Technology of China (UESTC) in 1983. He received a Ph.D. in electrical and computer engineering from Auburn University in 1997 and another Ph.D. in electrical engineering from Pennsylvania State University in 1998. From 1986 to 1989, he was a lecturer at the UESTC. From 1989 to 1993, he worked for the Technical University of Hamburg, Germany, working on microwave theory and RF designs. From 1997 to 2000, he worked for Hughes Network Systems of Hughes Electronics, Germantown, Maryland, where he was a member of technical staff in VLSI engineering, designing analog and digital ASICs for wireless and satellite communications. From 2000 to 2001, he worked for YAFO Networks, Hanover, Maryland, where he was a technical manager and a principal engineer in VLSI designs, leading high-speed SiGe IC designs for fiber communications. From 2001 to 2002, he worked for Cognio Inc., Gaithersburg, Maryland, designing RFICs for integrated multiband wireless transceivers. From 2002 to 2004, he was a RFIC consultant for Cognio Inc. In August 2002, he joined the faculty of Auburn University, where he is currently an associate professor in electrical and computer engineering. His research interests include VLSI circuits for digital, analog, and mixed-signal applications, RFIC designs for wireless and broadband communications, ultrahigh frequency synthesis, and analog/mixed-signal built-in self-test (BIST). Dr. Dai is a senior member of the IEEE and has served as a guest editor for the *IEEE Transactions on Industrial Electronics*. He currently serves on the technical program committee of the symposium on VLSI circuits.

Index

Recent Titles in the Artech House Microwave Library

Active Filters for Integrated-Circuit Applications, Fred H. Irons

Advanced Techniques in RF Power Amplifier Design, Steve C. Cripps

Automated Smith Chart, Version 4.0: Software and User's Manual, Leonard M. Schwab

Behavioral Modeling of Nonlinear RF and Microwave Devices, Thomas R. Turlington

Broadband Microwave Amplifiers, Bal S. Virdee, Avtar S. Virdee, and Ben Y. Banyamin

Classic Works in RF Engineering: Combiners, Couplers, Transformers, and Magnetic Materials, John L. B. Walker, Daniel P. Myer, Frederick H. Raab, and Chris Trask, editors

Computer-Aided Analysis of Nonlinear Microwave Circuits, Paulo J. C. Rodrigues

Design of FET Frequency Multipliers and Harmonic Oscillators, Edmar Camargo

Design of Linear RF Outphasing Power Amplifiers, Xuejun Zhang, Lawrence E. Larson, and Peter M. Asbeck

Design of RF and Microwave Amplifiers and Oscillators, Pieter L. D. Abrie

Distortion in RF Power Amplifiers, Joel Vuolevi and Timo Rahkonen

EMPLAN: Electromagnetic Analysis of Printed Structures in Planarly Layered Media, Software and User's Manual, Noyan Kinayman and M. I. Aksun

FAST: Fast Amplifier Synthesis Tool—Software and User's Guide, Dale D. Henkes

Feedforward Linear Power Amplifiers, Nick Pothecary

Generalized Filter Design by Computer Optimization, Djuradj Budimir

High-Linearity RF Amplifier Design, Peter B. Kenington

High-Speed Circuit Board Signal Integrity, Stephen C. Thierauf

Integrated Circuit Design for High-Speed Frequency Synthesis, John Rogers, Calvin Plett, and Foster Dai

Intermodulation Distortion in Microwave and Wireless Circuits, José Carlos Pedro and Nuno Borges Carvalho

Lumped Elements for RF and Microwave Circuits, Inder Bahl

Microwave Circuit Modeling Using Electromagnetic Field Simulation, Daniel G. Swanson, Jr. and Wolfgang J. R. Hoefer

Microwave Component Mechanics, Harri Eskelinen and Pekka Eskelinen

Microwave Engineers' Handbook, Two Volumes, Theodore Saad, editor

Microwave Filters, Impedance-Matching Networks, and Coupling Structures, George L. Matthaei, Leo Young, and E.M.T. Jones

Microwave Materials and Fabrication Techniques, Second Edition, Thomas S. Laverghetta

Microwave Mixers, Second Edition, Stephen A. Maas

Microwave Radio Transmission Design Guide, Trevor Manning

Microwaves and Wireless Simplified, Thomas S. Laverghetta

Modern Microwave Circuits, Noyan Kinayman and M. I. Aksun

Neural Networks for RF and Microwave Design, Q. J. Zhang and K. C. Gupta

Nonlinear Microwave and RF Circuits, Second Edition, Stephen A. Maas

QMATCH: Lumped-Element Impedance Matching, Software and User's Guide, Pieter L. D. Abrie

Practical Analog and Digital Filter Design, Les Thede

Practical RF Circuit Design for Modern Wireless Systems, Volume I: Passive Circuits and Systems, Les Besser and Rowan Gilmore

Practical RF Circuit Design for Modern Wireless Systems, Volume II: Active Circuits and Systems, Rowan Gilmore and Les Besser

Production Testing of RF and System-on-a-Chip Devices for Wireless Communications, Keith B. Schaub and Joe Kelly

Radio Frequency Integrated Circuit Design, John Rogers and Calvin Plett

RF Design Guide: Systems, Circuits, and Equations, Peter Vizmuller

RF Measurements of Die and Packages, Scott A. Wartenberg

The RF and Microwave Circuit Design Handbook, Stephen A. Maas

RF and Microwave Coupled-Line Circuits, Rajesh Mongia, Inder Bahl, and Prakash Bhartia

RF and Microwave Oscillator Design, Michal Odyniec, editor

RF Power Amplifiers for Wireless Communications, Steve C. Cripps

RF Systems, Components, and Circuits Handbook, Second Edition, Ferril A. Losee

Stability Analysis of Nonlinear Microwave Circuits, Almudena Suárez and Raymond Quéré

TRAVIS 2.0: Transmission Line Visualization Software and User's Guide, Version 2.0, Robert G. Kaires and Barton T. Hickman

Understanding Microwave Heating Cavities, Tse V. Chow Ting Chan and Howard C. Reader

For further information on these and other Artech House titles, including previously considered out-of-print books now available through our In-Print-Forever® (IPF®) program, contact:

Artech House	Artech House
685 Canton Street	46 Gillingham Street
Norwood, MA 02062	London SW1V 1AH UK
Phone: 781-769-9750	Phone: +44 (0)20 7596-8750
Fax: 781-769-6334	Fax: +44 (0)20 7630 0166
e-mail: artech@artechhouse.com	e-mail: artech-uk@artechhouse.com

Find us on the World Wide Web at: www.artechhouse.com