Signal Generation Using High-Order Delta-Sigma Modulation

by

Xavier Haurie, B. Eng. 1993

Department of Electrical Engineering

McGill University, Montréal

November 1996

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of Master of Engineering

0-612-29597-4

Abstract

This work presents high-order, arbitrary-band delta-sigma oscillators. They are a class of digital circuits which, augmented with a minimum of analog circuitry requiring no trimming, generate fully programmable, high-quality analog sinusoidal signals. A generalization of previous work, they can meet arbitrary signal-band and SNR specifications at a minimum digital hardware cost, and without the previously reported stability problems. It is shown that multitone generation requires but simple modifications to the basic oscillator topology; this signal generation scheme is thus highly attractive for endowing mixed-signal integrated circuits and systems with self-test capabilities. Delta-sigma oscillators can be useful in other applications as well.

An essential building block of delta-sigma oscillators is a one-bit digital delta-sigma modulator with unity Signal-Transfer-Function. A complete, computer-aided design method, relying on a novel high-order modulator topology allowing the use of power-of-two coefficients, is formulated and justified. Although the resulting modulators are aimed specifically at usage in delta-sigma oscillators, they can find applications in oversampled D/A conversion in general as they require a minimal amount of digital hardware.

DSMOD is the computer-aided design tool which was developed to automate the design, simulation and prototyping processes. It implements a number of involved design algorithms, and allows for a quick comparison of theoretical, simulated and prototype behavior, with the use of a graphical user interface. It is written mostly for MATLAB and is thus highly portable and expandable.

The measurements performed on prototypes prove the soundness, flexibility and efficiency of DSMOD. They also prove that low hardware cost and high performance levels are attainable with the novel delta-sigma modulator and oscillator topologies presented here.

Résumé

Cette thèse présente des oscillateurs delta-sigma de haut ordre et à bande de fréquence arbitraire. Il s'agit d'une classe de circuits numériques, utiles pour la génération d'ondes analogiques sinusoidales, entièrement programmables et de haute qualité. En tant que généralisations d'un circuit proposé antérieurement, ils permettent de satisfaire des spécifications arbitraires de la fréquence et du rapport signal-bruit des ondes générées, et ce à un coût minimal et sans problèmes de stabilité. Il est aussi démontré que la génération simultanée de multiples ondes sinusoidales ne requiert que de simples modifications à la configuration de base de l'oscillateur. Ce mode de génération de signal est donc très attrayant pour douer des circuits ou systèmes analogiques-numériques de la capacité d'autotest. Les oscillateurs delta-sigma peuvent aussi trouver une utilité dans d'autres applications.

Une composante essentielle d'un oscillateur delta-sigma est un modulateur delta-sigma à sortie 1 bit et dont la fonction de transfert du signal est unitaire. Une méthode de conception de ces circuits complète et informatisée est présentée. Elle est basée sur une nouvelle topologie utilisant des coefficients égaux à des puissances de deux. Bien qu'elle vise tout particulièrement l'application aux oscillateurs delta-sigma, cette méthode est jugée utile pour la conception de tout système de conversion numérique à analogique suréchantillonée.

DSMOD est l'outil de conception assistée par ordinateur développé afin d'automatiser cette méthode et d'évaluer ses résultats par des simulations et des prototypes. Ce logiciel est écrit pour MATLAB et est donc facilement portable et augmentable.

Les expériences réalisées sur des prototypes d'oscillateurs et de modulateurs prouvent la validité et l'efficacité des méthodes présentées dans cette thèse. Elles démontrent aussi les hauts niveaux de performance rendus possibles par les nouveaux circuits proposés.

Acknowledgments

First and foremost, I wish to express my deep gratitude to Dr. Gordon Roberts. His patience, his many encouragements, his unwavering enthusiasm for our research work and the opportunity he gave me to become part of the international design community were a blessing. His real concern and uncommon ability for teaching and sharing insight, together with the respect he shows his students, make him not just a supervisor but a true mentor.

Many warm thanks to my lab mates, in particular Michael Toner, Benoît Veillette, Jean-Paul Chaib, Evan Hawrysh, Mourad El-Gamal, Robert Wodnicki, Ara Hajjar, Nicholas Gagnon, John Abcarius, Loai Louis, Michael Kim, Palash Desai and so many others for much shared knowledge, insight and questioning, and for their friendship.

Special thanks to Ara Hajjar for his invaluable help with the FPGA prototypes, and to Arman Hematy for his help with the silicon prototype. Nicholas Gagnon and Lilian Zia gave me a much needed hand with a number of experiments and with the development of DSMOD.

The FCAR Fund and the MICRONET gave me the financial support without which I could not have completed this work. Together with NSERC and CMC they also funded most of the equipment which I used on a daily basis. Jacek Slaboszewicz did an impressive job at keeping the computers running around the clock.

There are no appropriate words to thank those who helped me most when the future seemed bleakest, and for their unconditional love and support: my family, and Tiza.

A la mémoire de mon grand-père, Edmond Haurie.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Self-Testable Mixed-Signal Systems

A substantial portion of the cost of manufacturing electronic circuits and systems is taken up by testing. The fundamental reason is that integrated-circuit fabrication exhibits significant random device parameter variations and as a result the working circuits must be sorted from the bad ones based on some test. In addition. testing is capital-intensive. as opposed to design which is almost purely knowledge-intensive. Finally. part of the product value resides in the confidence that the customer places in it. which a manufacturer can not guarantee solely by good design practices but necessarily also by testing.

Testing may be done on a system's components at each fabrication step and throughout its entire life-time. However the amount of testing done at each stage of the system's life depends on economic issues and cannot be worked out as a general rule. Another testing issue regards which tests are to be performed by external devices. and which are to be part of self-test schemes. There are many factors pushing manufacturers to adopt self-test strategies for their products. One is that external testers are extremely expensive. and that the circuits must be tested sequentially. so that the test duration has a significant impact on the cost of a circuit. Self-testable circuits need only be powered-up and instructed to self-test and to report the result. which requires a fraction of the capital immobilized in a full-fledged tester. Another factor is the need for system-level diagnostic tests for servicing installed equipment. Finding the failed component in a complex system is costly if an operator or technician is assigned to the job. Components which signal the host system when they no longer meet specifications thus reduce maintenance costs (the trade-off being increased component complexity and cost). Yet another motivation for self-testability is the technical difficulty of bringing signals in and out of the device under test over a significant length of connector (a few feet), especially at high frequencies (for instance at the 1 GHz range used in wireless communications), a problem which is not as acute in a self-test scheme. The proper way to view built-in self-test is as an investment in hardware, or even a reuse of existing hardware, which returns profits whenever the device or system must be tested.

Requirements for self-testability should thus be incorporated in the system specifications so as to reflect choices made according to the economics of development, fabrication and maintenance of its components, and also according to the level of confidence in the final product required by the consumer. Such issues have been dealt with for a variety of purely digital circuits such as memories and microprocessors, and digital Built-In-Self-Test (BIST) is now widespread [1]. The same can not be said of mixed-signal circuits and systems (i.e. containting both digital and analog circuits), mainly because analog self-test remains a technical challenge.

Mixed-signal circuits play an increasingly important role in the microelectronics industry. In an overwhelming number of applications, electronic systems are designed to accept analog electric signals generated by sensors (e.g. microphones, light detectors, acceleration detectors, receiving antennas), or to provide analog outputs to "actuators" (e.g. speakers, displays, electric motors, transmission antennas), or both. Although they interface with analog signals, much of the processing needed in such systems is performed by digital circuits (hence the name of *mixed-signal* systems) which offer more reliability, precision and insensitivity to manufacturing variations than their analog counterparts. In some cases the analog components may be reduced to as little as analog-to-digital and digital-to-analog converters (ADC's and DAC's) but some analog circuitry will always be required.

The fundamental hurdle in testing the functionality of a mixed-signal system is that it involves generating or reading analog signals, according to a scheme which exposes the devices not meeting specifications. The researcher's role is then to develop design principles and paradigms for self-testability, applied to mixed-signal systems, and to expose the design trade-offs which are involved so that engineers have tools enabling them to meet the self-testability specifications. Any proposal for endowing mixed-signal circuits with self-testing capability will be doubly attractive if it integrates into the existing digital BIST schemes and reuses existing digital hardware.

A number of such principles have been proposed recently, such as the Mixed-Analog-Digital Built-In-Self-Test (MADBIST) scheme [2][3], and more are being developed, such as the IEEE 1149.4 standard [4] which defines a chip- and system-level analog test-bus. The present work deals with one component of the MADBIST scheme, namely the analog sinewave source, in an attempt to generalize and improve previous work done in that field [5][6][7][8] and to automate the design of this crucial circuit in the emerging area of mixed-signal self-test. This work also makes possible the use of high-quality on-chip signal sources in other, unforeseen applications.

1.2 Literature Review on Analog Sinewave Generation

One key principle of the MADBIST is that the analog source is a mostly digital circuit and can thus be tested by a standard digital BIST before it is used to excite the mixed-signal and analog circuits under test. The literature on analog signal generation is reviewed next, in light of this particular requirement.

1.2.1 Analog Resonators

Purely analog oscillators [9] are not reviewed in any detail here because they are hard to integrate reliably, as argued in [5]. Some reasons are that integrated inductors needed in passive implementations have poor quality factors and are prohibitively large, and crystals are not integratable at all. Active implementations of analog oscillators using opamps display limited precision and are affected by device parameter variations. Integrated precision analog components require trimming, which adds to the fabrication cost. Finally, in a mixed-signal self-test scheme, the circuit generating the test stimulus must be tested in the first place. It is obvious that using an analog implementation for the signal source does not get one very far in this regard. Signal generation methods relying on digital circuits combined with digital-to-analog conversion are reviewed instead.

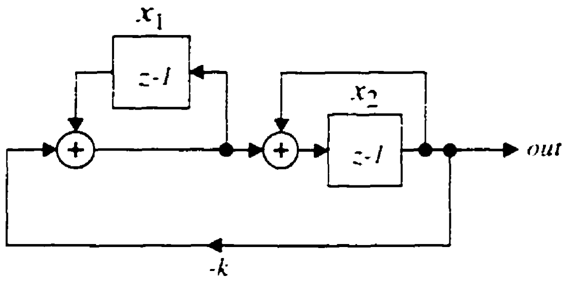


Fig. 1.1: **Lossless Discrete Integrator (LDI) resonator.**

1.2.2 Direct Digital Frequency Synthesis

Direct Digital Frequency Synthesis or DDFS, as used in [10] for example, consists in storing in a ROM the values of a normalized sinusoid taken at a large number of evenly-spaced time-instants. An indexing variable (representing the instantaneous phase of the signal) stored in an accumulator cycles through the ROM addresses with a chosen phase increment. The shortcomings of this method are that frequency selectivity comes at the price of a large number of ROM words, while decent SNR performance entails a large word-length, making for a significant area overhead. In addition, arbitrary signal amplitude requires that the output of the ROM be scaled, another costly operation. Moreover, the ROM cannot be implemented by reusing digital hardware. All in all the hardware cost can be prohibitive for high-precision signal generation. Finally, a high-precision digital-to-analog converter is needed. In a mixed-signal self-test scheme this converter would not be testable and would thus impair the scheme's reliability.

1.2.3 LDI Resonator

LDI-resonators and their implementation are discussed to great length in [11]. An LDI-resonator such as the one shown in Fig. 1.1 generates a digital sinusoidal signal based on its registers' initial conditions and an input parameter k. It consists of two discrete-time integrators which hold the resonator's state variables $x_1(n)$ and $x_2(n)$, and a multiplier which implements the scaling-by-k operation. Provided that $0 < k < 4$, the output $x_2(n)$ is a sinusoidal signal described by the following equation:

$$x_2(n) = A \sin(\Omega_0 n + \phi), \qquad (1.1)$$

where the normalized angular frequency Ω_0, the amplitude A and the phase ϕ of the sinusoid depend on the loop coefficient k and the initial conditions $x_1(0)$ and $x_2(0)$, and are given by:

$$\Omega_0 = \operatorname{acos}\left(1 - \frac{k}{2}\right), \qquad (1.2)$$

$$A = \frac{(1-k)x_2(0) + x_1(0)}{\sin(\Omega_0 + \phi)}, \qquad (1.3)$$

and

$$\phi = \operatorname{atan}\frac{x_2(0)\sin\Omega_0}{(1 - k - \cos\Omega_0)x_2(0) + x_1(0)}. \qquad (1.4)$$

Since three independent variables (k, $x_1(0)$ and $x_2(0)$) are available to control the circuit's operation, the three parameters of the sinusoid (A, Ω_0, and ϕ) can independently be set to arbitrary values. For instance, to obtain a period of oscillation of T samples and an amplitude A, the loop coefficient and the initial values of the two integrator variables x_1 and x_2 can be set according to:

$$k = 2\left(1 - \cos\left(\frac{2\pi}{T}\right)\right), \qquad (1.5)$$

$$x_2(0) = A, \qquad (1.6)$$

$$x_1(0) = \frac{k}{2}A. \qquad (1.7)$$

As argued in [5], the LDI resonator, due to its simplicity and programmability, is an attractive alternative to a ROM-based digital sinusoidal generator, as long as the multiplier implementing the scaling factor k is kept simple. The resolution of the generated signal depends on the number of bits used in the implementation of each of the circuit's blocks. However, just like DDFS, this signal generation scheme implies the use of a high-

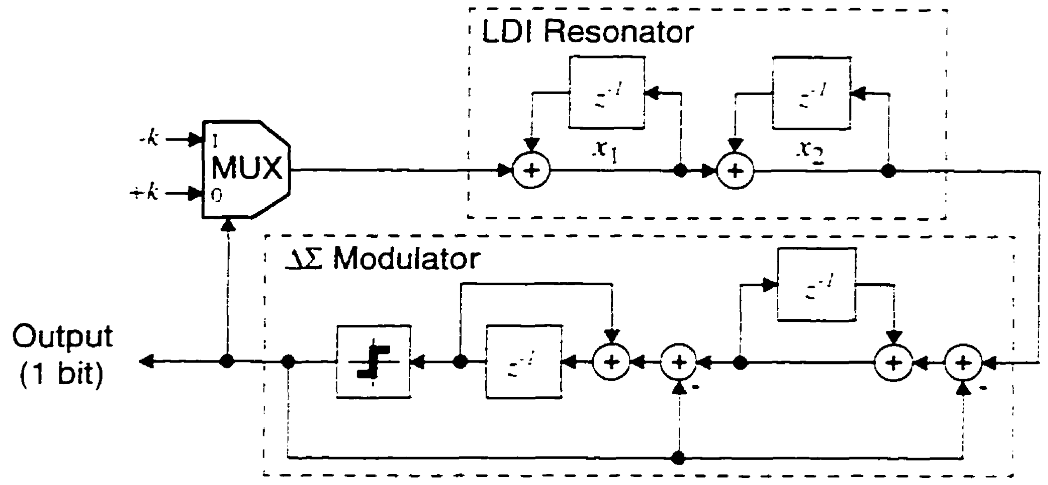


Fig. 1.2: **2nd-order delta-sigma oscillator.**

resolution digital-to-analog converter, and again this solution, although potentially more economical than DDFS, is not entirely suitable for use in a mixed-signal self-test scheme.

1.2.4 Delta-Sigma Oscillator

The delta-sigma oscillator was proposed in [5] to overcome the need for a multiplier and a high-resolution DAC while retaining all the advantages of analog signal generation based on an LDI-resonator. This circuit consists of an LDI-resonator, a 2nd-order delta-sigma modulator and a 2-input multiplexer, as shown in Fig. 1.2. We thus refer to it as a 2nd-order delta-sigma-oscillator. Its operation is described next.

In order to explain the operation of the delta-sigma oscillator, we first take a look at how the digital sinusoid generated by the LDI-resonator of Fig. 1.1 could be converted to a continuous-time analog signal. This can be achieved with a minimal amount of analog hardware using a digital delta-sigma modulator (this circuit will be discussed in detail in Chapter 2), a one-bit DAC and a linear analog filter as shown in Fig. 1.3. The input to the modulator must be highly oversampled (i.e. the sample rate must be significantly larger than the signal bandwidth), which allows the modulator to compute a one-bit representation of the signal. The input digital signal is encoded in the average of the modulator output over some significant number of samples, hence the requirement for oversampling. The one-bit DAC is used to create an analog equivalent of the modulator output, which is

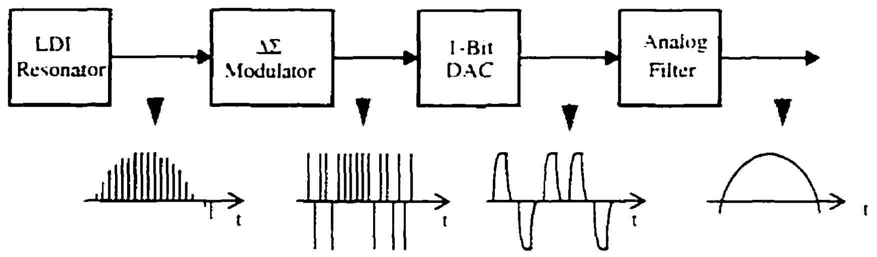


Fig. 1.3: Analog sinewave generation using an LDI-resonator and a delta-sigma modulator.

then smoothed by a lowpass filter to produce a highly accurate analog counterpart to the digital input. This process is described in more detail in Section 2.1.

The delta-sigma oscillator of Fig. 1.2 is an attractive variant on the scheme shown in Fig. 1.3. The modulator has been inserted in the resonator loop: as it introduces a unit delay the integrators in the resonator have been modified to take this into account. The justification for inserting the modulator in the resonator is that at low frequencies the modulator reproduces its input faithfully, while whatever noise it generates at high frequencies will be attenuated by the resonator integrators. If the programmable scaling-by-k operation is done on the modulator output then it can be cheaply implemented by a simple 2-way multiplexer, instead of a full-fledged multiplier. This is possible because the modulator outputs a one-bit signal. The generated signal is embedded in the modulator output bit stream and is obtained with the use of a one-bit DAC and some analog lowpass filtering, just as in Fig. 1.3. Note that the amplitude of the generated sinusoidal signal must be limited to slightly less than the modulator's reference level, otherwise the oscillator will become unstable due to non-linear effects.

The inband power density spectrum for one possible delta-sigma oscillator output is shown in Fig. 1.4. This plot shows a bandwidth equal to one 128th of the digital sample rate. The spectral plots were obtained by running a 2^{18}-point FFT on simulated data and averaging over 8 bins. The generated sinusoid is represented by the sharp power-density spike at the frequency 0.002 x F_S. Its power is roughly -55 dB relative to the output level of the delta-sigma modulator, equivalent to an amplitude of 0.5. The modulator produces the spectrally-shaped noise floor whose total inband power is equal to -75 dB relative to the modulator output level. The signal-to-noise-power ratio (SNR) for this simulation is

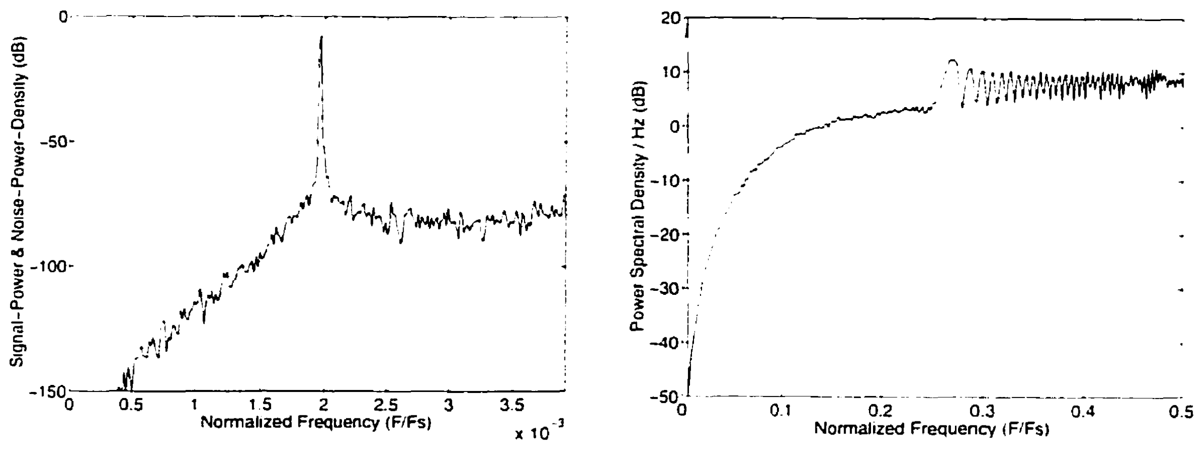


Fig. 1.4: Power spectral density of a signal generated by a 2nd-order delta-sigma oscillator.

thus 70 dB. Over a smaller bandwidth, the SNR figure would be larger, and conversely if the bandwidth were increased. The maximum possible SNR is fixed for a given ratio of the digital sampling rate to the signal bandwidth, according to a relationship provided in [5].

The 2nd-order delta-sigma oscillator meets the MADBIST requirement of using mostly digital hardware. Some analog filtering is also required, but is assumed to be present in the circuit under test. If not this filtering operation can be implemented by a cheap RC-network with very loose specifications (i.e large tolerances on the component values and matching). No trimming is required on these analog components. Because it is built mostly using digital hardware, this signal source is fully and accurately programmable, impervious to process and temperature variations, and as stated above, self-testable by the standard methods applicable to digital circuits. It can reuse digital hardware such as adders and registers already present in the system, or even transform a digital tester into an analog tester with the help of a programmable analog filter.

Since [5] a number of improvements have been proposed to delta-sigma oscillators. They are: multitone signal generation [6][6][7], bandpass delta-sigma oscillators and their use in high-frequency signal generation [12] and more specifically in a proposed MADBIST for wireless systems [13], FM-signal generation [14], and high-order delta-sigma oscillators [16]. All of these topics except the latter two are also treated in [15]. The present thesis deals essentially with high-order and arbitrary-signal-band oscillators, as well as their computer-aided design [18]. The resulting improvements are listed next.

1.3 Proposed Improvements

There are some important limitations to the 2nd-order delta-sigma oscillator, which require explanations and solutions. The object of this thesis is to present a general, automated design framework which addresses these limitations.

1.3.1 Arbitrary Signal Band Location

In many applications signals occupy an arbitrary frequency band within the Nyquist interval, as in the case of communications systems in which the signal is modulated and occupies a given portion of the frequency spectrum. Thus there is a need for signal sources capable of generating clean tones near some arbitrary center frequency in $[0, F_S/2]$, i.e. bandpass or highpass oscillators. This is addressed both in Chapter 2, where bandpass and highpass modulators are discussed, and in Chapter 3, where it is shown how to set the center frequency of the signal band to an arbitrary value.

1.3.2 Arbitrary Stable Bandwidth

There is a limit to the bandwidth over which the oscillator can generate a stable tone. This has been verified experimentally for the 2nd-order oscillator in [5] and [15]. This stability problem can be explained by the fact that the delta-sigma modulator noise constitutes an additional input which can throw the oscillator out of its intended limit cycle. The solution here is to use a slightly more complex oscillator topology in which the amount of noise-injection can be controlled, as first proposed in [16]. The resulting oscillators are described in Chapter 3, along with a study of their stability properties.

1.3.3 Arbitrary SNR

Another limitation is the introduction of noise by the delta-sigma modulator in the signal band. This noise limits the signal-to-noise ratio of the generated tone. If the OSR is defined as the ratio of half the sampling rate to the signal bandwidth, then the inband noise power is fixed for a given value of the OSR for a 2^{nd}-order delta-sigma oscillator. For a given digital clocking rate, the oscillator may not be able to produce tones with a high SNR over a sufficient bandwidth. The solution resides in using a modulator of higher order which keeps the oscillator loop critically stable while creating a lower noise-floor. Such modulators were first proposed in [17] and are presented in detail in Chapter 2: they are designed to be implemented using a minimal amount of digital hardware.

1.3.4 Design Automation

The present work explains how the trade-offs between bandwidth, stability, SNR and hardware cost can be addressed. However the design techniques which are presented, although they are simple to understand, are computationally involved. To demonstrate their applicability, a computer-aided-design (CAD) tool had to be developed [18]. It is presented in Chapter 4.

Chapter 2

Low-Cost One-Bit High-Order Digital Delta-Sigma Modulators

2.1 Introduction

Delta-sigma modulation has become a technique of choice for implementing reliable and accurate integrated data converters for narrowband signals. This technique makes high-precision data-conversion possible with the use of a coarse DAC (for D/A conversion) or ADC (for A/D conversion [19]) and digital and analog filters. Because it is required that the signal to be converted occupy a small bandwidth relative to the sample rate, we speak of oversampling data conversion. The oversampling ratio (OSR) is defined as the ratio of half the sampling rate F_s to the signal bandwidth BW:

$$OSR = \frac{F_s/2}{BW}. \tag{2.1}$$

A lowpass oversampling D/A converter using a single-bit DAC and a digital delta-sigma modulator is depicted in Fig. 2.1. It consists of three stages: the digital delta-sigma modulator having a one-bit output, followed by the one-bit DAC operating at the digital sample rate, and a lowpass analog filter with corner frequency at the signal band edge.

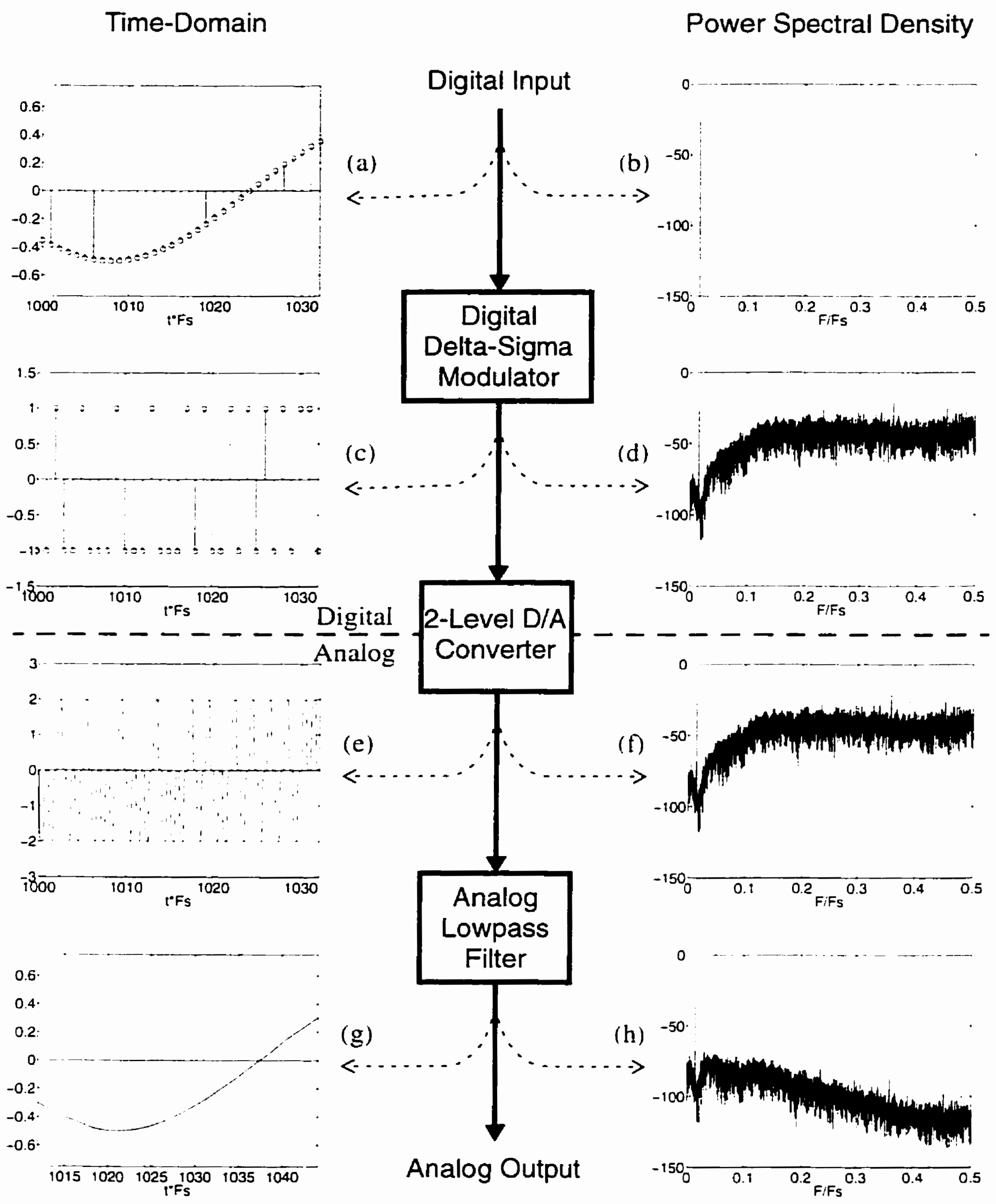


Fig. 2.1: Example of an oversampling digital-to-analog converter. On the left are shown the signals in the time-domain. On the right the same signals are represented in the frequency domain.

On the left of the block diagram are shown the input and output of each block in the time domain, on the right in the frequency domain. The input of the modulator is the digital signal to be converted to the analog domain. In the present example it is a high-resolution sinusoid whose frequency is much smaller than the sample rate F_s. The modulator outputs a stream of +1 and -1 values whose local density equals its instantaneous input. This fundamental aspect of the delta-sigma modulation process is better illustrated in the spectral density plot shown in Fig. 2.1(d): the low-frequency input signal is very faithfully reproduced while large amounts of noise are introduced at higher frequencies. Note that the modulator does introduce some low-power noise in the signal band.

The single-bit DAC then creates an analog signal corresponding to the stream of +1's and -1's generated by the modulator: each digital bit coming from the modulator translates into a finite-duration pulse, with non-zero rise-time and fall-time. Finally, the analog lowpass filter attenuates the high-frequency noise introduced by the modulator. As a result of this filtering operation, the fast-changing pulse-train coming from the 2-level DAC is smoothed to an accurate, continuous-time, continuous-valued version of the digital sinusoidal input.

Note that our example showed a lowpass D/A conversion system; a similar converter for bandpass signals is realized by using a bandpass delta-sigma modulator, i.e. one that introduces very little noise over some midband range of frequencies, and a bandpass analog filter. Likewise, highpass D/A conversion systems can be built from a highpass delta-sigma modulator and a highpass analog filter. Oversampling A/D converters function in an analogous manner, using a sampled-data analog modulator, a coarse ADC and a digital filter.

The major advantage of oversampling D/A conversion over its Nyquist-rate counterpart is that it works with a low-resolution, high-speed "Nyquist-rate" DAC, even though the conversion itself is highly accurate. The trade-off is that digital-signal-processing hardware is required to implement the delta-sigma modulator and that only bandlimited signals can thus be converted. For bandlimited, integrated applications this trade-off is more than acceptable, since high-resolution "Nyquist-rate" D/A converters are difficult to fabricate reliably, whereas digital-delta-sigma modulators, just like other digital-signal-processing devices, are more easily and reliably integrated. Moreover, the resolution of the conversion can be increased either by using higher-order delta-sigma modulation or by increasing the OSR, without necessitating better analog component precision.

The contributions to the field of delta-sigma modulation which are the most significant to the present work, specifically for one-bit delta-sigma D/A conversion, are reviewed in Section 2.2. The beauty of delta-sigma modulation is that it replaces the problem of component matching and accuracy encountered in the design of Nyquist-rate converters with a special kind of transfer-function design problem. Indeed, even though it is a non-linear device, the functionality of a delta-sigma modulator can be accurately modelled and designed as that of a linear system described by two transfer functions, which are defined in Section 2.3. The constraints posed by the specific application of delta-sigma modulation to delta-sigma oscillators are the object of Section 2.4. Section 2.5 presents a complete method for designing high-order, arbitrary-signal-band modulators for use in delta-sigma oscillators with a minimal amount of digital hardware, based on an appropriate choice of modulator topology and on the idea of coefficient quantization. The issues of simulation and prototyping are also addressed. Finally, conclusions are drawn in Section 2.6.

2.2 Literature Review on Delta-Sigma Modulation

A comprehensive review of delta-sigma modulation techniques used in oversampling A/D converters is presented in [20]. This is relevant here because many principles and modulator architectures for A/D conversion can be reused in the context of oversampling D/A conversion. A slightly older publication [21] sums up some of the most important contributions to the field of delta-sigma modulation since the 1960's, both for D/A and A/D conversion. To the best of our knowledge, delta-sigma modulation was first applied to D/A conversion, as opposed to A/D conversion, in [22]. A high-order D/A delta-sigma modulator using a single-bit DAC was reported as early as 1987 in [23]. One-bit modulators work best at large oversampling ratios (at least 16, sometimes larger than 100) and thus require significant amounts of digital processing power. However their advantage lies in the simplicity of the two-level DAC used in conjunction with them. In fact a one-bit DAC is an inherently linear device and thus introduces less harmonic distortion than multi-bit DACs, even without trimming. The DACs presented in [24] are good examples of such relatively simple analog circuits which can be used in high-resolution D/A conversion systems. A large portion of the literature focuses on one-bit modulators. In particular, the maximum SNR achievable by one-bit modulators of orders 1 to 8 is given in [25] for various values of the oversampling ratio

Nonetheless, oversampling data converters using multi-bit DACs are often deemed a good compromise between one-bit, highly-oversampled converters and Nyquist-rate converters [26]. The main reason is that they grant the benefits of noise-shaping while improving the stability properties of the modulator and keeping its complexity low. Some delta-sigma modulator architectures which necessitate a multi-bit DAC are multi-stage or MASH modulators [27], time-interleaved modulators [28] and parallel modulators [29]. It has recently been shown that the error created by the non-idealities of a multi-bit DAC can be whitened and even spectrally-shaped out of the signal band [30], making multi-bit delta-sigma modulation very attractive for both D/A and A/D conversion. This technique requires only a reasonable amount of additional digital hardware. This is a recent development and no experimental results are available to support it yet. Therefore the present thesis deals exclusively with one-bit delta-sigma modulators. Also, in certain applications the use of a single-bit DAC and hence modulator is preferable, in particular when the complexity of the analog circuitry must be kept to a strict minimum, such as for mixed-analog-digital built-in-self-test [3] and oversampled analog signal generation [6][8][16]. There is thus a need for a method to design one-bit digital delta-sigma modulators with a minimum amount of digital hardware. Providing such a method is the goal of this chapter.

Delta-sigma modulation has recently been generalized to include bandpass [31] modulators, which, together with highpass modulators, are given consideration in the present work. Other further extensions of delta-sigma modulation, such as complex modulators [32], are not treated here. Most of the work presented in this chapter was first introduced in [17].

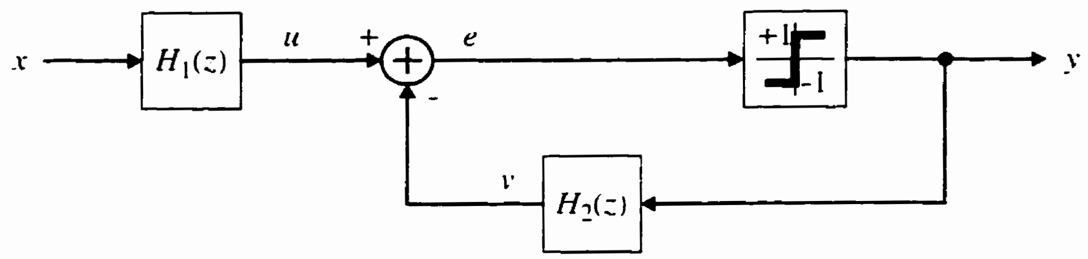


Fig. 2.2: **Signal-flow graph of a general high-order delta-sigma modulator.**

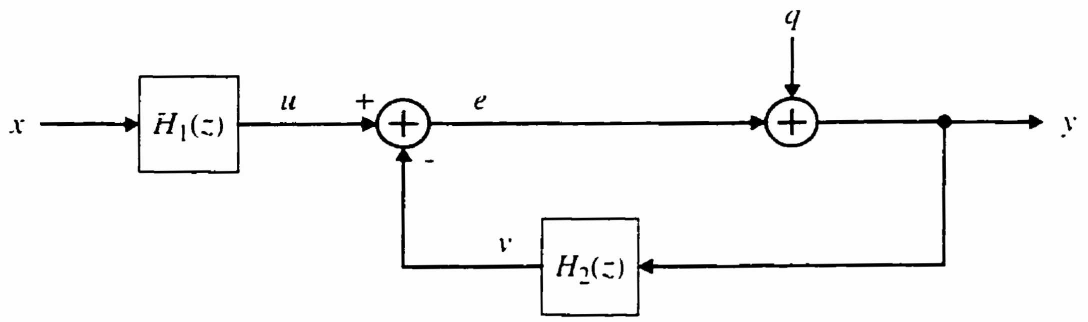


Fig. 2.3: **Linear model of the general high-order delta-sigma modulator.**

2.3 Linear Model of Delta-Sigma Modulation

Without loss of generality, it may be assumed that any high-order one-bit delta-sigma modulator can be represented by the block-diagram shown in Fig. 2.2. It consists of three main blocks: two linear filters H_1 and H_2 and a one-bit (or two-level) quantizer. The bandlimited input signal, x, is filtered by H_1 and the feedback signal from H_2 is subtracted from the result to produce e, the input to the quantizer. The output of the quantizer is fed directly to the modulator output y. It is also filtered by H_2 to create the feedback signal. A delta-sigma modulator is thus a non-linear system with feedback.

The quantizer can be modelled as an additive noise-source so as to obtain a linear model of the delta-sigma modulator, as shown in Fig. 2.3. Specifically, the quantizer has been replaced by a summer whose inputs are the quantizer input e and a noise source q, and whose output is the quantizer output y. The noise input q is equal to the quantization error made at each sampling instant by the 2-level quantizer. As explained in [19], it is then straightforward to relate the output y to the signal input x and to the noise input q in the frequency domain (the uppercase notation is used to indicate the Z-transform of a signal):

$$Y(z) = \frac{H_1(z)}{1+H_2(z)} \cdot X(z) + \frac{1}{1+H_2(z)} \cdot Q(z). \tag{2.2}$$

We define the signal-transfer-function $STF(z)$ and the noise-transfer-function $NTF(z)$ as follows:

$$STF(z) = \frac{H_1(z)}{1+H_2(z)}. \tag{2.3}$$

$$NTF(z) = \frac{1}{1+H_2(z)}. \tag{2.4}$$

The spectral behavior of any delta-sigma modulator (even those which do not have the form of the block diagram shown in Fig. 2.2) can then be fully described by these two transfer functions. That is, the Z-transform of the output y is written as:

$$Y(z) = STF(z) \cdot X(z) + NTF(z) \cdot Q(z). \tag{2.5}$$

$Q(z)$, the Z-transform of q, is assumed to be white noise. The incentive for making this assumption is that multi-bit quantizers are traditionally modelled as additive white noise sources. In general, this model is far from valid for a one-bit quantizer, but it turns out that it is accurate enough when applied to high-order delta-sigma modulators, as verified below from simulation results. From Eqn. (2.2), it is clear that if the magnitude of $H_2(z)$ is large in the signal band, the noise input q will be attenuated. At these frequencies, $H_1(z)$ must be correspondingly large for the signal x to be passed unaltered to the output y. In other words we want the STF to be close to 1 and the NTF close to 0 in the signal band.

The linear model given by Eqn. (2.5) is not complete unless the Power Spectral Density (PSD) of $Q(z)$ is estimated. For multi-bit quantizers, this PSD is taken to be white noise with average power equal to $P_Q = \frac{\Delta^2}{12}$, where Δ is the size of the quantization interval. However this approximation is only valid because the quantization error is practically evenly distributed in any quantization interval, regardless of the signal level, which is not the case for a one-bit quantizer.

Luckily, for all practical purposes the quantization noise can still be assumed to be white in high-order modulators, but the average quantization error power P_Q depends on the signal level and is derived in another fashion. First, let us have the following definitions. The PSD of the modulator input $X(z)$ is defined as $S_X(f) = |X(z)|^2|_{z = e^{j2\pi f}}$, and the PSD of the quantization error $Q(z)$ is assumed to be a constant, i.e. independent of frequency, which we denote as S_Q. The signal power is then given by $P_X = \int_0^{f_s} S_X(f)df$, and the quantization error power by $P_Q = \int_0^{f_s} S_Q df = f_s S_Q$, where f_s is the digital sample rate. Assuming the signal and the quantization noise to be uncorrelated, the output PSD is derived from Eqn. (2.5) to be:

$$S_Y(f) = S_X(f)|STF(e^{j2\pi(f/f_s)})|^2 + S_Q|NTF(e^{j2\pi(f/f_s)})|^2. \qquad (2.6)$$

Integrating Eqn. (2.6) over frequency yields the following expression for the output power:

$$P_Y = \int_0^{f_s} S_X(f)|STF(e^{j2\pi(f/f_s)})|^2 df + \int_0^{f_s} S_Q|NTF(e^{j2\pi(f/f_s)})|^2 df. \qquad (2.7)$$

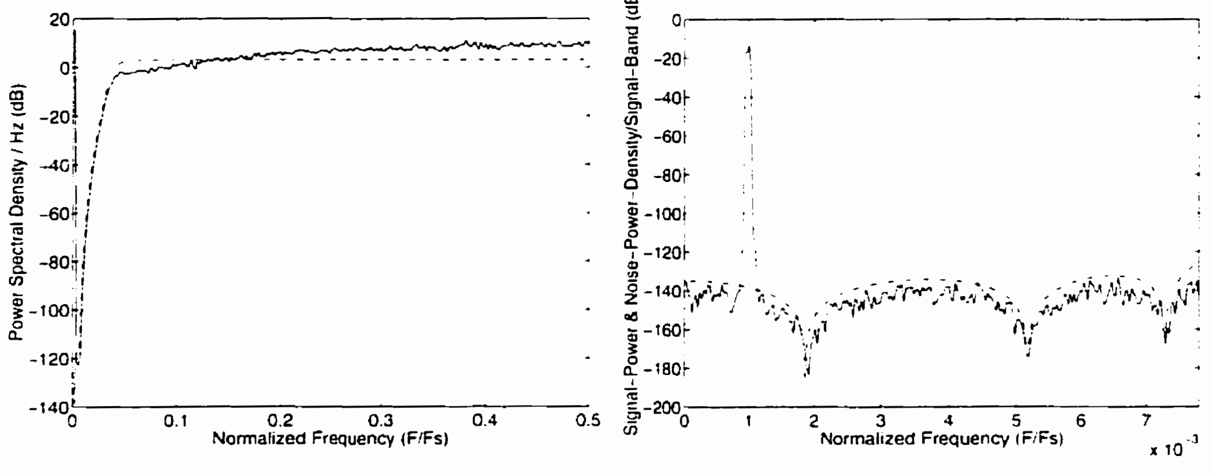
Since the output y is always +1 or -1, its power P_Y is equal to 1. Note also that the modulators used in delta-sigma oscillators have $STF(e^{j2\pi(f/f_s)}) = 1$ (this is explained in Section 2.4.2). Eqn. (2.7) thus simplifies to

$$1 = P_X + S_Q \cdot \left(\int_0^{f_s} |NTF(e^{j2\pi(f/f_s)})|^2 df\right), \qquad (2.8)$$

or equivalently,

$$S_Q = \frac{1 - P_X}{\int_0^{f_s} |NTF(e^{j2\pi(f/f_s)})|^2 df}, \qquad (2.9)$$

where $\int_0^{f_s} |NTF(e^{j2\pi(f/f_s)})|^2 df$ is the average value of the NTF-magnitude-squared on the unit circle, scaled by f_s. This last equation gives a good estimate of the actual quantization



(a) (b)

Fig. 2.4: (a) Nyquist-band and (b) Inband power spectrum for a 6th-order modulator. The solid line represents simulation data, while the dotted line represents the noise spectrum predicted by the linear model.

noise power density. We finally obtain the following estimate of the modulator output's PSD by using this expression for S_Q in Eqn. (2.6):

$$S_Y(f) = S_X(f) + \frac{(1-P_X)\left|NTF(e^{j2\pi(f/f_s)})\right|^2}{\int_0^{f_s}\left|NTF(e^{j2\pi(f/f_s)})\right|^2 df}. \tag{2.10}$$

If the input x is a sinusoid with amplitude A, then its average power is $\frac{A^2}{2}$ and the PSD of the output's noise component is $\dfrac{\left(1-\frac{A^2}{2}\right)\left|NTF(e^{j2\pi(f/f_s)})\right|^2}{\int_0^{f_s}\left|NTF(e^{j2\pi(f/f_s)})\right|^2 df}$.

The linear model of the output spectrum predicted by Eqn. (2.10) has been verified by simulating a 6th-order modulator designed as per the method of Section 2.5.7. The simulated and predicted spectra are represented in Fig. 2.4, by a solid and dotted line, respectively. It is clear from these plots that the linear model yields a very accurate estimate of the noise power density, both within the signal band and over the Nyquist interval. Note that in these two plots and in all subsequent spectral plots, the signal is shown as power, whereas the noise is shown as power density per signal band[1]. The SNR is simply read as the ratio of the signal level (-10 dB) to the average noise level (-140 dB), that is, 130 dB. Note also the absence of any perceptible harmonics of the signal tone, proof of the high linearity of one-bit delta-sigma modulators, despite the presence of the highly non-linear one-bit quantizer in the circuit.

To summarize, a one-bit delta-sigma modulator takes as input an "oversampled" multi-bit bandlimited signal and uses the extra bandwidth to encode the input in a stream of single bits. The one-bit output consists of an accurate replica of the input signal plus quantization noise which has been "spectrally shaped" so as to be greatly attenuated in the signal band.

2.4 Special Requirements for Use in ΔΣ Oscillators

As explained in Chapter 1 and in Chapter 3, the modulators which can be used in a delta-sigma oscillator have two special characteristics. These are reiterated and treated below.

2.4.1 One-Bit Quantizer

In the context of mixed-signal self test, delta-sigma oscillators must have a single bit output so as to minimize the complexity of the analog circuitry to a one-bit DAC and a linear filter. Thus the modulator used in the oscillator must use a one-bit quantizer. As a result, stability of the modulator is a non-trivial issue, and it will be dealt with in Section 2.5.1.

1. Let BW be the signal bandwidth, f_s/N the resolution bandwidth of the N-point FFT (i.e. the size of one FFT bin), and p_i the noise power found in the i^{th} FFT bin. Then the noise power density per signal band for that bin is defined as $p_i \cdot \frac{BW \cdot N}{f_s}$. The total inband noise power is easily obtained as the average of the inband noise power density per signal band.

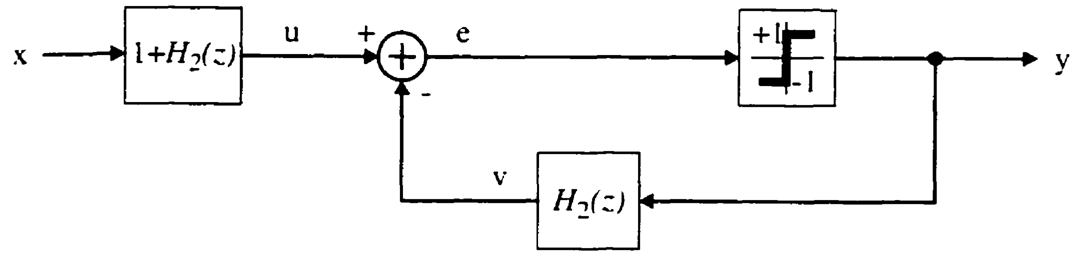


Fig. 2.5: Signal-flow graph of a general high-order delta-sigma modulator with unity STF.

2.4.2 Unity Signal Transfer Function

It will be demonstrated in Section 3.2 that the modulator must have a STF strictly equal to unity in order to be used in a delta-sigma oscillator. Here we present an overall modulator topology with that property.

From Eqns. (2.3) and (2.4), it is clear that if the linear filters H_1 and H_2 can be made to realize arbitrary transfer functions then the STF and NTF of the modulator of Fig. 2.1 can be set arbitrarily and independently of each other. Indeed, the NTF depends on H_2 only and to obtain a desired STF one can set H_1 according to:

$$H_1(z) = \frac{STF(z)}{NTF(z)} = STF(z)\,(1 + H_2(z))\,. \qquad (2.11)$$

To make the STF equal to unity, we need $H_1(z) = 1 + H_2(z)$, which is realized by the signal-flow graph shown in Fig. 2.5.

One can transform this signal-flow graph so that a single block $H(z)$ is used to realize both the feedback gain $-H_2(z)$ and the feedforward gain $1{+}H_2(z)$, instead of two distinct blocks as in Fig. 2.5. One such overall modulator structure guaranteeing a unity STF and using a single instance of $H(z)$ is presented in [33] and reproduced in Fig. 2.6. Here the quantization error is fed back as e while any feedback of the input signal x cancels out. Thus the STF is unity. By direct analysis, the NTF is found to be:

$$NTF(z) = 1 - H'(z) = 1 - \frac{H(z)}{1 + H(z)} = \frac{1}{1 + H(z)}\,. \qquad (2.12)$$

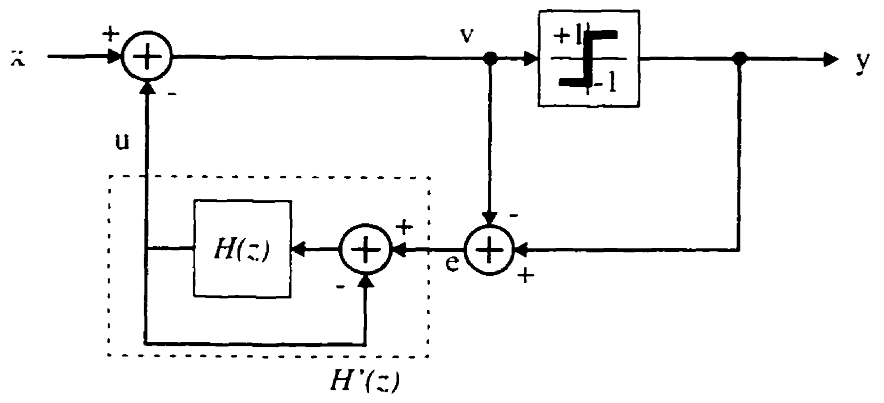


Fig. 2.6: An overall structure for delta-sigma modulators with unity Signal Transfer Function.

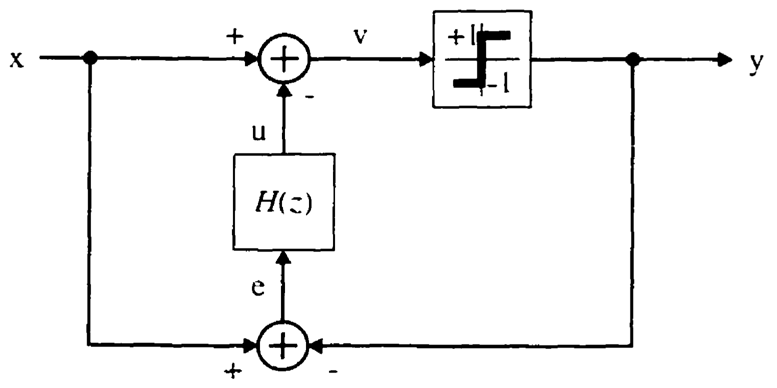


Fig. 2.7: Proposed alternative overall structure for delta-sigma modulators with unity Signal Transfer Function. The poles of the linear block H(z) are also the zeros of the Noise Transfer Function, which is a desired property of the modulator structure.

Another topology can be obtained by a straightforward manipulation of the signal-flow graph of Fig. 2.6, yielding the modulator topology shown in Fig. 2.7. When the quantizer is replaced by its linear model, the input signal feeds straight to the output and back to the filter input; it also feeds forward to the filter input with the opposite sign, and so the signal feedback is cancelled, resulting in a unity STF. The NTF is again equal to $\frac{1}{1+H(z)}$. This topology requires one less subtractor than the one of Fig. 2.6. We thus use it instead.

Since we have shown that the STF can be set to 1 independently of the NTF, the rest of this chapter will focus on the design and the implementation of the NTF, or equivalently, of the linear filter $H(z)$.

2.5 Design Method

A complete method by which a modulator is designed so as to convert high-precision signals to one bit over a given portion of the Nyquist interval with a given signal-to-noise power ratio (SNR) is presented here. The resulting modulators are meant to be used specifically in oscillator applications and we describe how to design them using as little hardware as possible.

This method comprises three main steps. First an NTF of the appropriate order, OSR and inband noise rejection is computed. Then a modulator topology is chosen and its coefficients are computed so that the desired NTF is realized. Finally, these coefficients are quantized to powers-of-two or canonical-signed digits (i.e. sums or differences of a few powers-of-two) so that they can be implemented more cheaply than using multipliers. This last process results in an actual NTF which departs from the desired one, but for carefully chosen modulator topologies this departure is kept to a minimum.

2.5.1 Noise-Transfer-Function Design

Rather than directly designing the transfer function $H(z)$, we focus instead on the noise transfer function. It is designed to attenuate the quantization noise enough in the band of interest so as to produce the desired SNR. If the passband type is bandpass or highpass, a lowpass prototype with the same oversampling ratio is designed first. Section 2.5.3. explains how the bandpass or highpass modulator is obtained from the realization of the lowpass prototype. We are thus faced with a pole-zero placement problem, albeit one with special constraints.

The first of these special constraints has to do with stability. In particular, since delta-sigma modulators are circuits with a highly non-linear element (the one-bit quantizer), their stability cannot be predicted using the methods for linear systems only. Fortunately, conditional stability can be guaranteed by an empirical criterion expressed in the frequency domain. An absolute requirement for stability, given in [34], is that the magnitude of the NTF be less than 2.0 at all frequencies, i.e.,

$$max(|NTF(e^{j2\pi f})|_{0<f<1/2}) < 2.0\,. \qquad (2.13)$$

As documented in [25], simulations show that the higher this maximum, the smaller the range of input amplitudes for which the modulator is stable. However, as the bound decreases toward 1.0, the NTF magnitude in the signal band increases and so does the inband noise power. The bound on the NTF which results in the maximum SNR is thus some value between 1.0 and 2.0. Finding the bound yielding the maximum SNR involves trying successive values of the bound, and for each of them finding the input level yielding the maximum SNR, through simulation. This process requires repetitive simulations and can be performed by the computer program presented in Chapter 4. Note that the traditional stability criterion for discrete-time linear systems, namely that the poles must lie within the unit circle in the Z-plane, also applies.

The second constraint is that the modulator be realizable. Recall that all feedback loops in a realizable discrete-time system must have at least a unit delay. In the case of the modulator of Fig. 2.7, the transfer function *H(z)* must be strictly causal. This implies that the denominator of *H(z)* is of higher order than the numerator, which in turns implies that the NTF is strictly proper, i.e. that its numerator and denominator have identical orders and leading coefficients. Formally, we write:

$$NTF(z)|_{z=\infty} = 1\,. \qquad (2.14)$$

In order to maximize the SNR, the zeros of the NTF, which are also the poles of *H(z)*, should lie on the unit circle within the signal band. The optimal zero locations for a given OSR are given in [25] for lowpass modulators. The NTF poles can be designed using specialized filter-design software such as the one presented in [33]. Alternatively, a more generally available tool such as MATLAB can be used instead to design a Butterworth, Chebychev or Elliptic pole configurations for the NTF (our CAD tool does just that). In all cases one can refer to the data in [25] to obtain the NTF order which is required to meet the SNR specification for the given OSR.

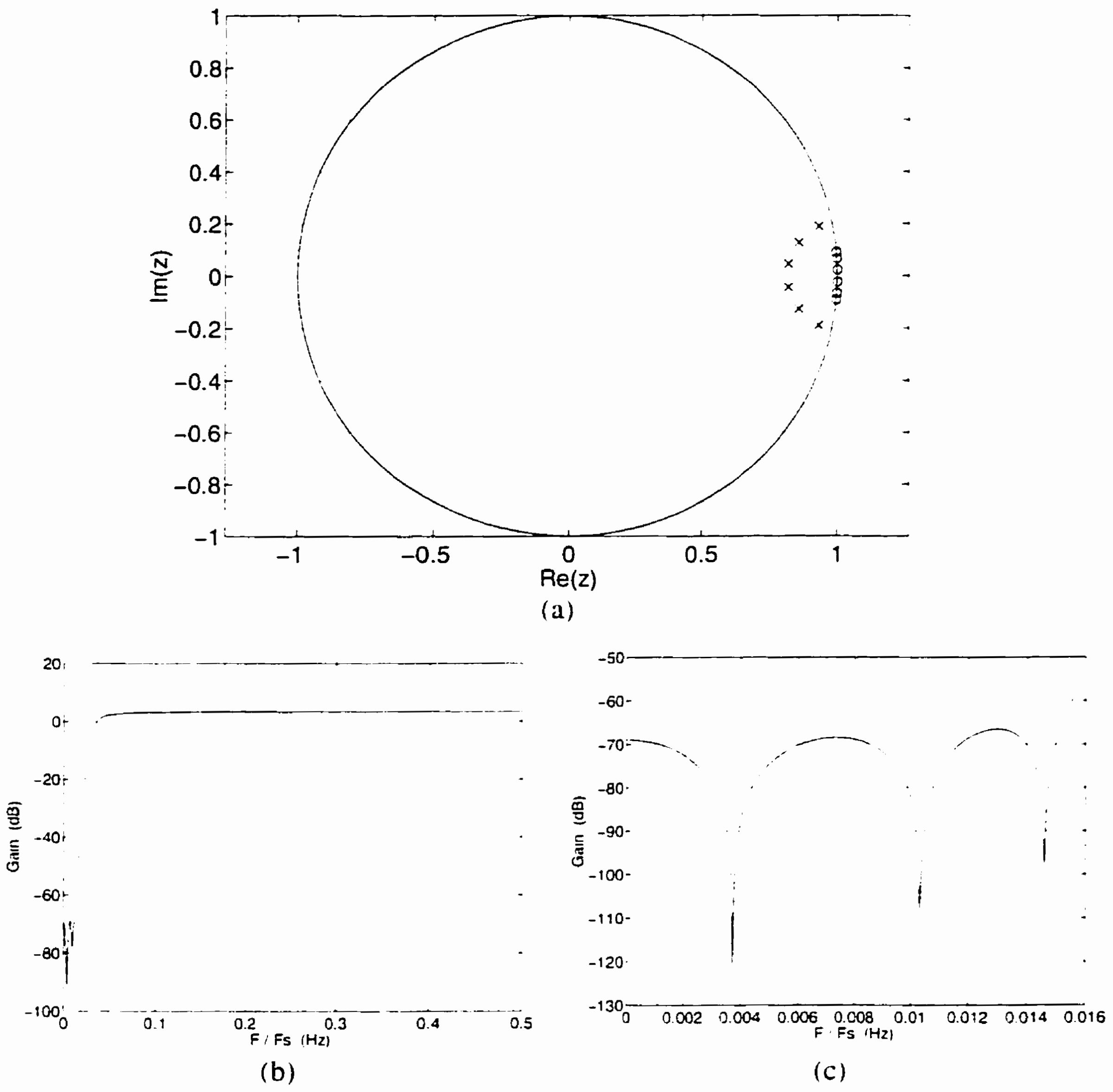


Fig. 2.8: (a) Pole-zero configuration of a 6th-order noise transfer function designed for an oversampling ratio of 32. (b) Magnitude of the NTF over the Nyquist interval; (c) in the signal band.

As examples, two 6th order NTFs, one for an OSR of 32 and the other for an OSR of 128, were designed using the same method as in [25]. The bound on the NTF magnitude was arbitrarily set to 1.5 which in general is a good first approximation in designing a modulator with optimal performance. Fig. 2.8(a) shows the pole-zero configuration of the NTF for an OSR of 32. Note the poles in a butterworth configuration, inside the unit circle, around the signal band. The zeros are on the unit circle at low frequencies, within the signal band. Fig. 2.8 (b) and (c) show the magnitude of the same NTF. As expected from the pole-zero configuration, the NTF magnitude is very small in the signal band (low frequencies). The 3 pairs of zeros create the three notches seen in the magnitude plot. At high frequencies the magnitude reaches a maximum of approximately 3 dB (corresponding to a gain of 1.5). Fig. 2.9 shows the same information for the NTF designed for an OSR equal to 128.

2.5.2 Modulator Topologies

We now turn to the problem of realizing a modulator with a given NTF of arbitrary order. We explore a variety of modulator structures, and will later evaluate which ones give good results with quantized coefficients, in Section 2.5.4. The three structures we will retain for that purpose are the resonator cascade, the integrator cascade, and the lossless-discrete-integrator ladder, or LDI ladder for short. They are discussed below.

The problem of realizing a modulator amounts to finding the structure coefficients which yield the desired NTF. For this, each term in the numerator and denominator of the NTF is expressed as a function of the structure coefficients, and these equations are solved simultaneously. Solving these non-linear equations is computationally intensive, and this is one of the reasons for which the entire design method presented here was coded into a computer-aided-design tool presented in detail in Chapter 4. The remainder of this section describes the modulator topologies under consideration.

A now widely-used high-order delta-sigma modulator realization was first introduced in [34] and [35] and is shown in Fig. 2.11. Here the realization of $H(z)$ consists of a cascade of integrators, with feedback applied from each integrator output to the input to set the poles. Feedforward branches to the output set the zeros. We thus call this structure the "integrator cascade with feedback and feedforward branches", or "integrator cascade" for short. Variants of this structure have since been used in successful switched-capacitor implementations [36][37] as well as digital ones [24]. One can modify the structure of Fig. 2.11 to give a unity STF by using the overall topology of Fig. 2.7, as shown in Fig. 2.11.

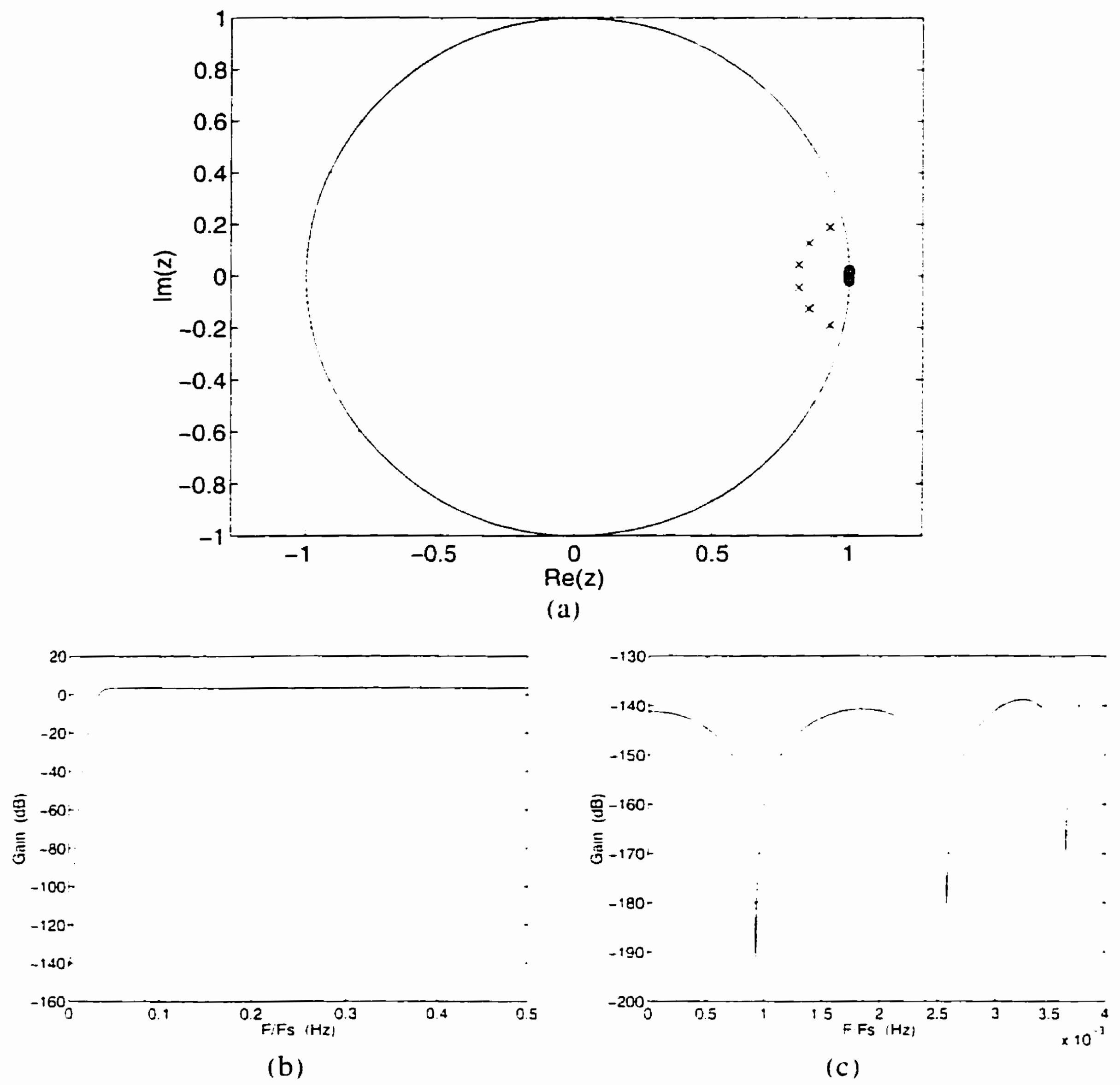


Fig. 2.9: (a) Pole-zero configuration of a 6th-order noise transfer function designed for an oversampling ratio of 128, with magnitude bounded by 1.5. (b) Magnitude of the NTF over the Nyquist interval; (c) in the signal band.

Another modulator topology, used in [24] and reproduced in Fig. 2.12, is often used to realize digital delta-sigma modulators. This is due to the fact that the B coefficients are multiplying single-bit values; these operations are cheaply implemented by two-input multiplexers instead of multipliers. However this topology's STF is not unity. The structure can be modified to have unity STF as shown in Fig. 2.12, but then the B coefficients

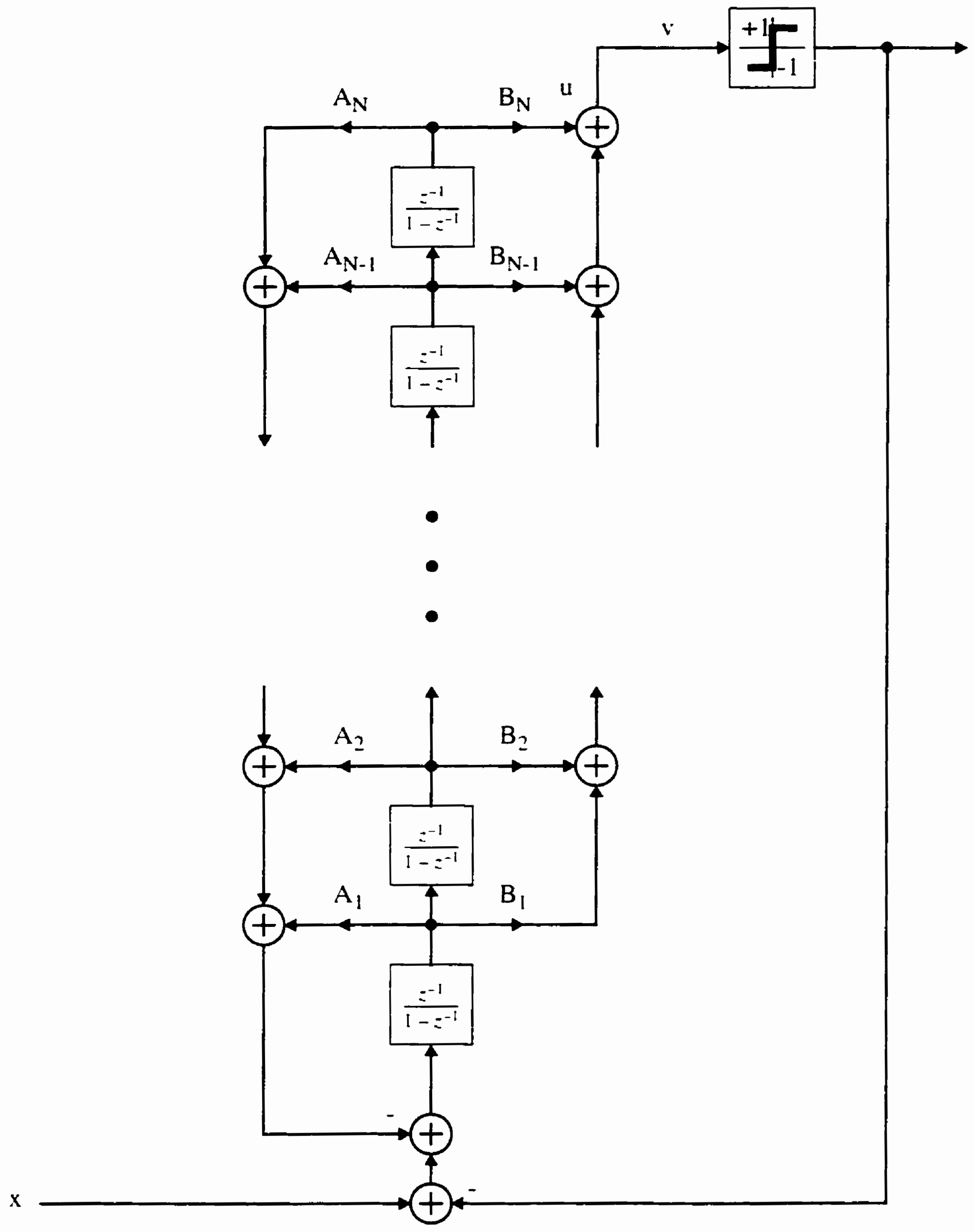


Fig. 2.10: Realization of an Nth-order delta-sigma modulator using Chao's multiloop-feedback integrator structure.

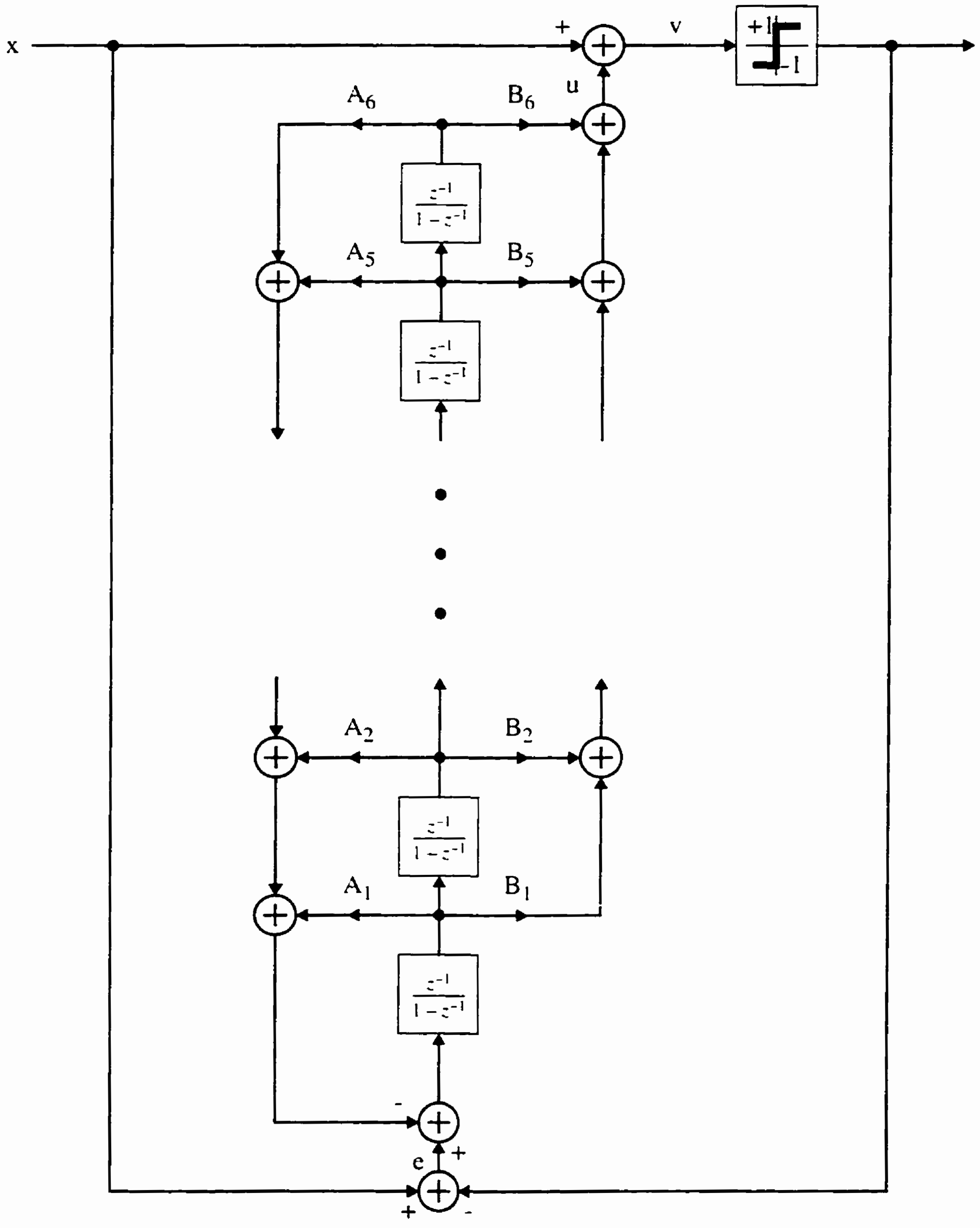


Fig. 2.11: Realization of an N^{th}-order delta-sigma modulator based on Chao's multiloop-feedback integrator structure, using the overall structure resulting in a unity signal transfer function.

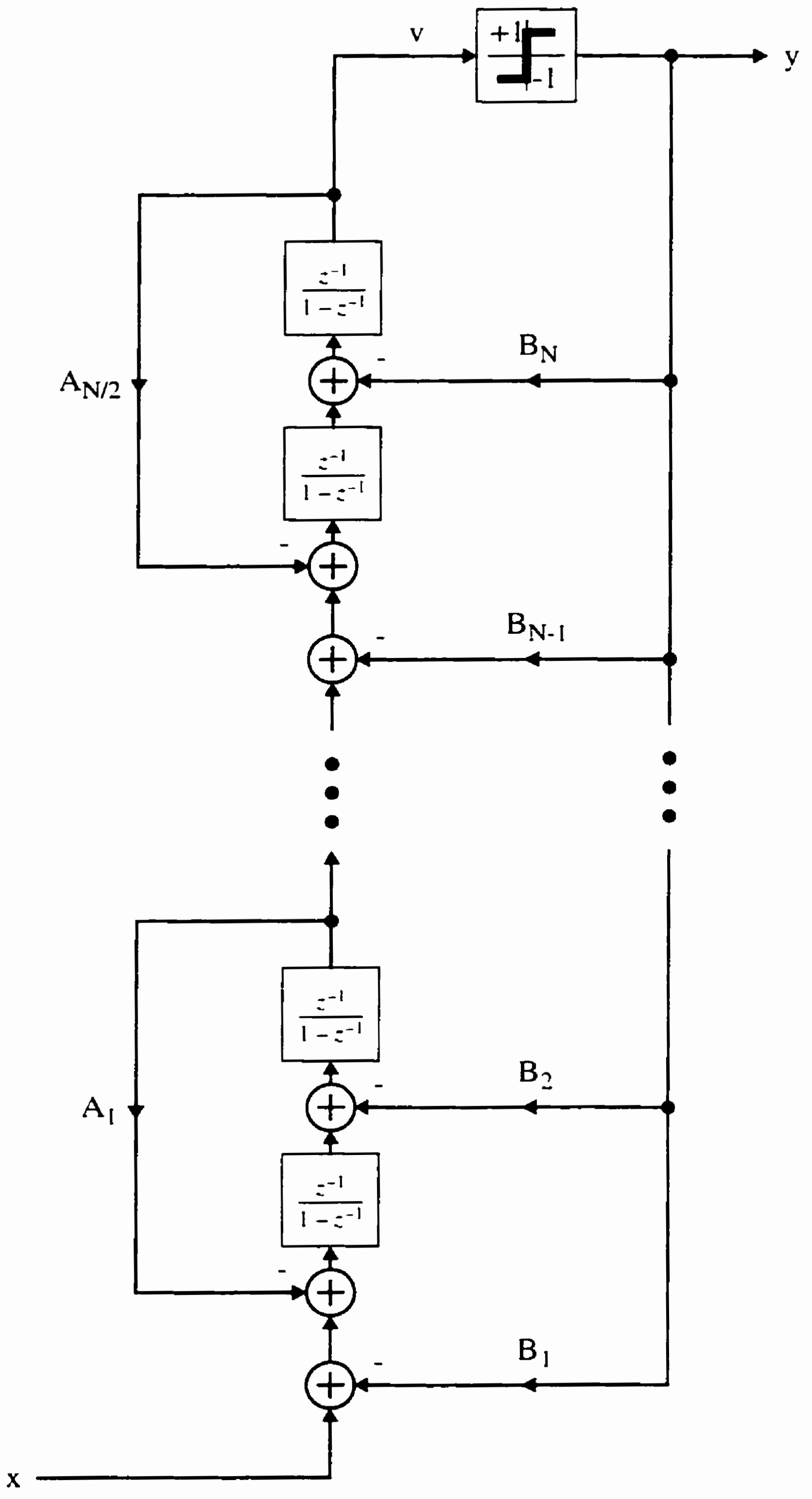


Fig. 2.12: Realization of an Nth-order delta-sigma modulator based on [24]. Note that the STF is not unity in this case.

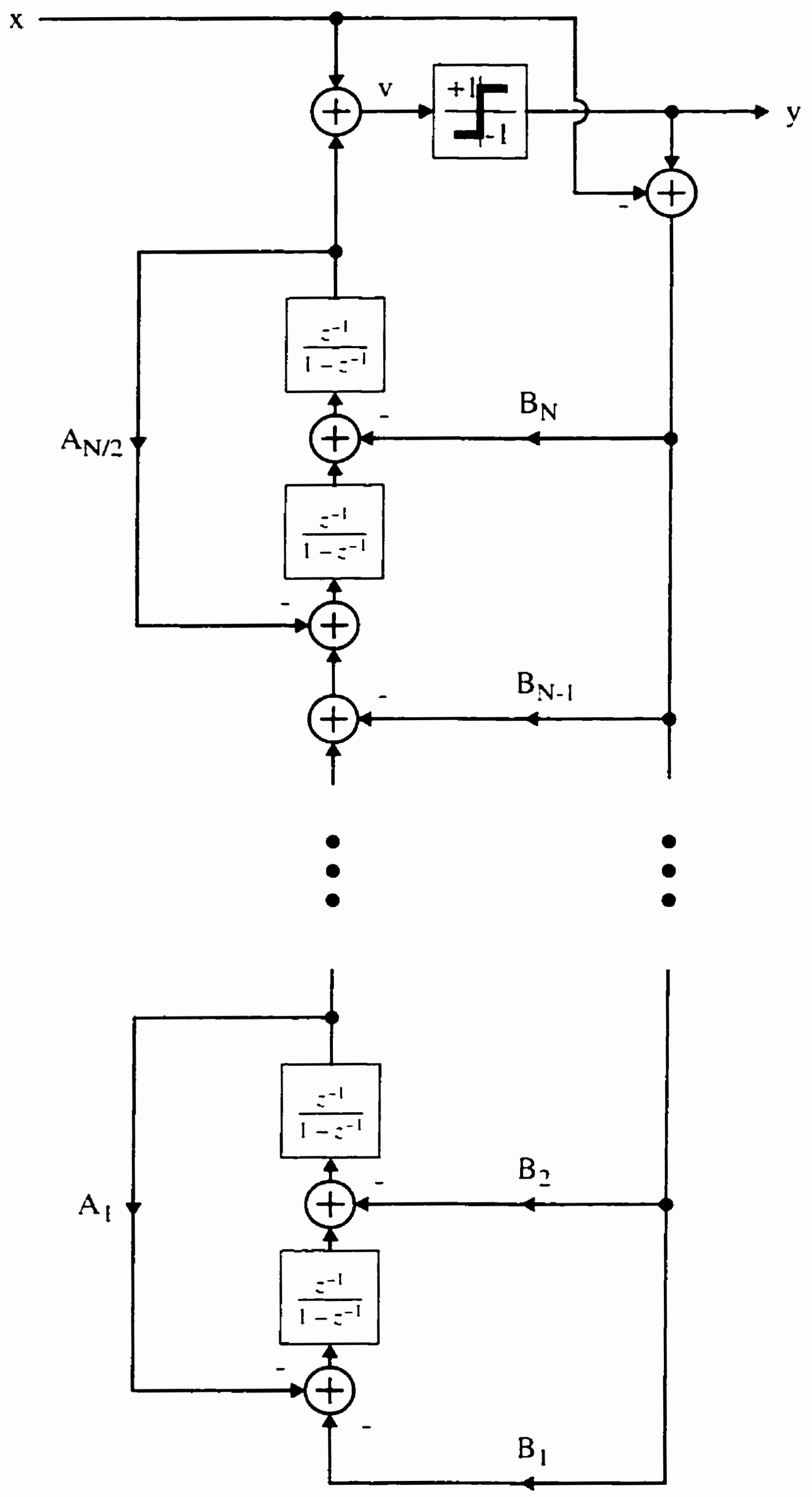


Fig. 2.13: Realization of an N^{th}-order delta-sigma modulator based on [24], but modified to have unity STF.

must be implemented by multipliers. Alternatively the B coefficients can be placed in another manner as shown in Fig. 2.15, and we call this particular realization a resonator-cascade. Note that in this particular structure the forward and backward integrators are interleaved and grouped two-by-two as resonators. In this case also we can no longer play the trick of implementing the B coefficients using multiplexers because they are used to scale a multi-bit signal instead of a single-bit one. This in turn is a result of the STF being equal to unity.

Finally, an essential contribution of this work is an LDI-ladder structure, similar to the ladder structures used to realize high quality analog filters because of their superior sensitivity properties. LDI-ladder filters were introduced in [38]; the modulator LDI-ladder structure we propose is shown in Fig. 2.15. It is composed of discrete-time integrators. The output of each integrator is first scaled by a coefficient A_i and then is fed forward to the next integrator in the ladder as well as back to the previous one (except for the first and last integrators). Each integrator output is also further scaled by a coefficient B_i and fed forward to the quantizer input. Note that this structure displays the main characteristic of ladders, that is, the way the feedback is distributed around the integrators, and for that reason we expect it to have low sensitivity to coefficient variations. In general, LDI ladders are designed from a continuous-time prototype which is mapped to a discrete-time equivalent, as described in [38]. However here we use a more direct approach to obtain the realization coefficients. Namely, we solve the non-linear equations relating the coefficients to the NTF terms, as for the integrator and resonator cascade structures.

2.5.3 Passband Mapping

In the case when the signal band is not centered around DC, the modulator structures presented in Section 2.5.2 must be modified to implement a bandpass or highpass NTF. In general, any linear system can be transformed from lowpass to bandpass or highpass by mapping the Z-variable to a new variable [39]. For a bandpass modulator with

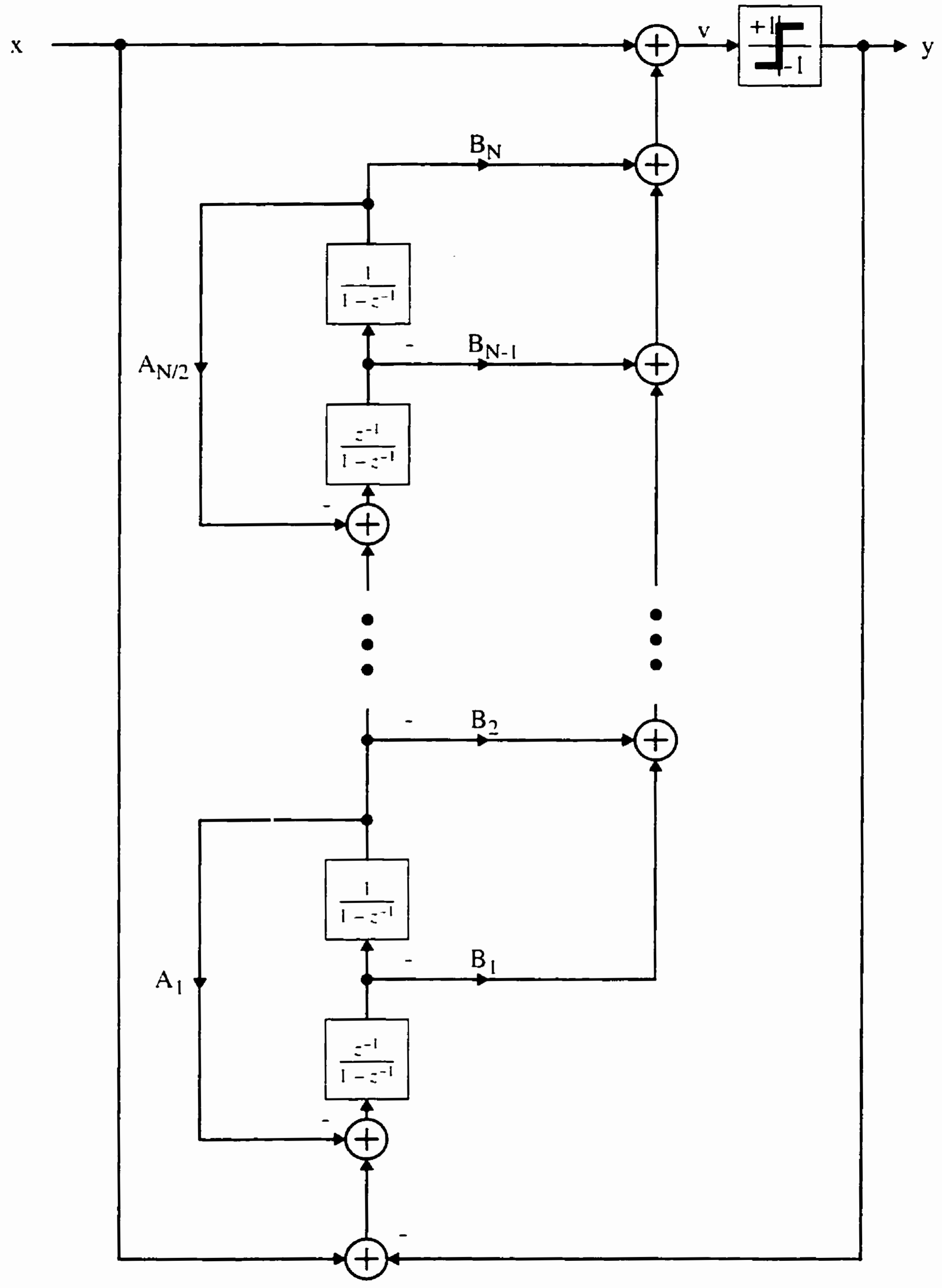


Fig. 2.14: Realization of an N^th-order modulator based on a resonator cascade, with unity STF.

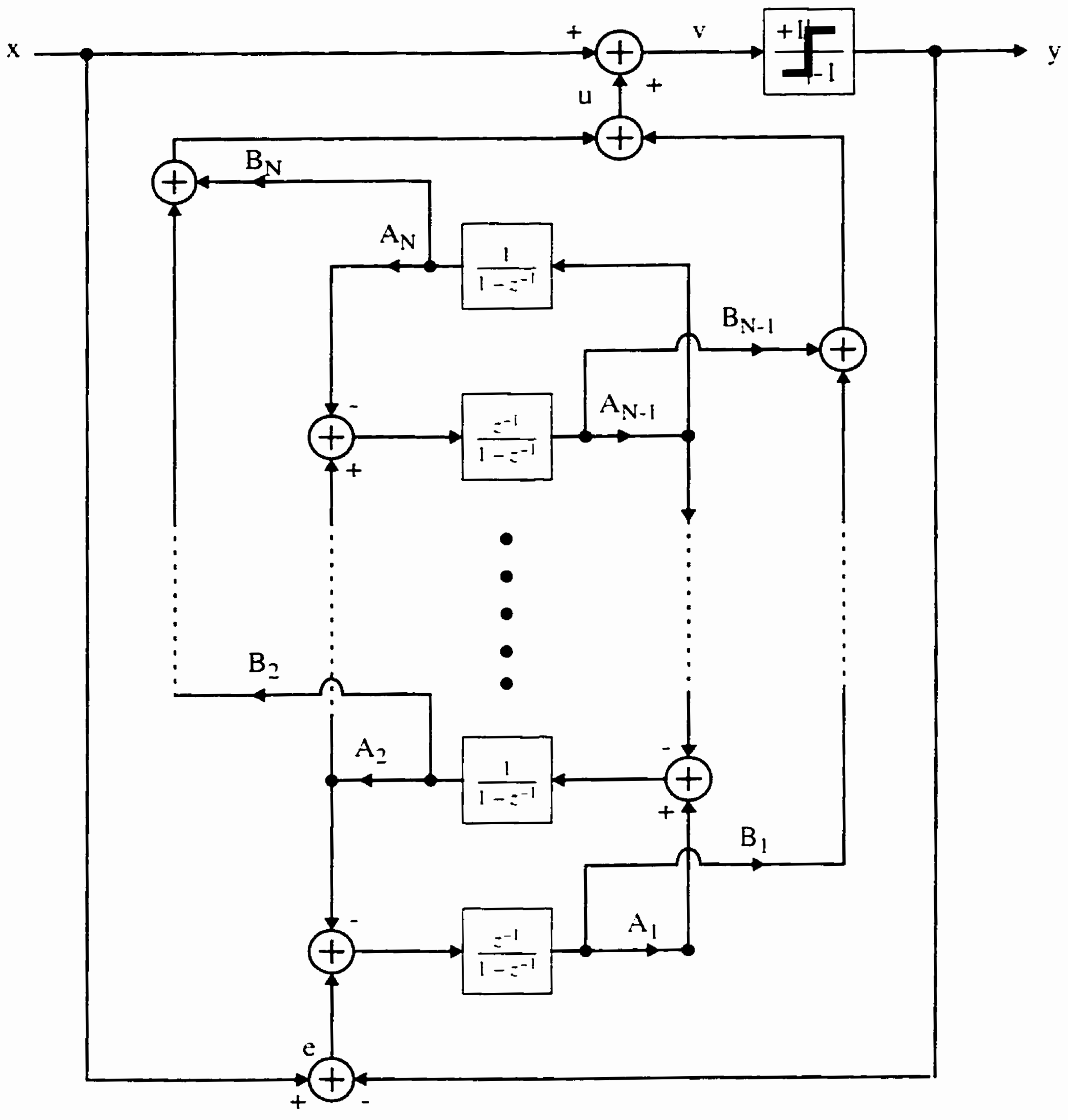


Fig. 2.15: **Realization of an N^th^-order modulator based on the Lossless Discrete Integrator (LDI) ladder, with an overall structure yielding a signal transfer function equal to unity.**

center frequency ω_c and an OSR defined as the ratio of the signal bandwidth to half the sampling rate, the mapping is described by:

$$z^{-1} \rightarrow -\frac{z^{-2} - Cz^{-1}}{-Cz^{-1} + 1}. \tag{2.15}$$

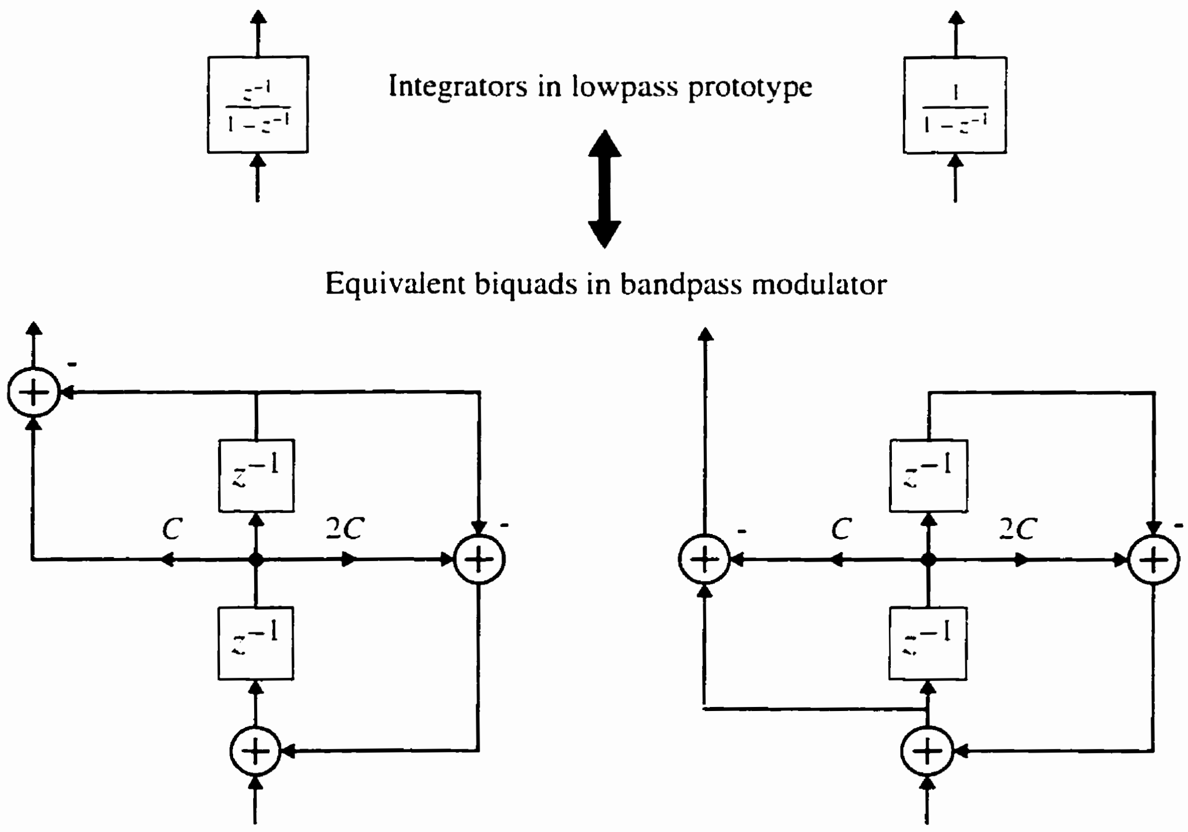

Integrators in lowpass prototype
$\frac{z^{-1}}{1-z^{-1}}$
$\frac{1}{1-z^{-1}}$
Equivalent biquads in bandpass modulator
z^{-1}
C
$2C$
-

Fig. 2.16: Integrator-to-biquad mappings for bandpass modulators.

where

$$C = -\frac{\cos\omega_c}{\cos\left(\frac{\pi}{2\cdot \mathrm{OSR}}\right)}. \tag{2.16}$$

In terms of modulator realization, once C has been found, the integrators in the lowpass prototype realization must be replaced by the appropriate biquads, shown in Fig. 2.16 and described by the following equations:

$$\frac{z^{-1}}{1-z^{-1}} \rightarrow -\frac{z^{-2}-2Cz^{-1}}{z^{-2}-2Cz^{-1}+1} \tag{2.17}$$

and

$$\frac{1}{1-z^{-1}} \rightarrow \frac{-Cz^{-1}+1}{z^{-2}-2Cz^{-1}+1}. \tag{2.18}$$

One very interesting case arises when the center frequency ω_c is equal to $\pi/2$. In that case C is 0 and the Z-variable mappings reduces to:

$$z^{-1} \rightarrow z^{-2}, \tag{2.19}$$

which amounts to doubling each delay element in the circuit.

Finally, to obtain a highpass modulator, the mapping is given by:

$$z^{-1} \rightarrow -z^{-1}. \tag{2.20}$$

The integrators then map according to:

$$\frac{z^{-1}}{1-z^{-1}} \rightarrow -\frac{z^{-1}}{1+z^{-1}} \tag{2.21}$$

and

$$\frac{1}{1-z^{-1}} \rightarrow \frac{1}{1+z^{-1}}. \tag{2.22}$$

2.5.4 Coefficient Quantization

The modulators realized with any of the topologies presented in Section 2.5.2 are not suited for an economical digital implementation, because they require multipliers to implement the scaling coefficients. However, if an approximation of the desired NTF can be found with all coefficients of the realization equal to powers-of-two, then the modulator can be implemented using only adders, subtractors, registers and fixed-shift units. The following section exposes our strategy for coming up with such an efficient realization, and the results for each of the topologies described above.

After the NTF has been designed and the corresponding modulator coefficients have been obtained, a method inspired from the one presented in [40] is used to quantize these coefficients to powers-of-two (including the biquad coefficients in a bandpass modulator). Each coefficient in the structure is rounded either upward or downward to the nearest power-of-two, independently of the other coefficients. For a structure with n coefficients, this gives rises to 2^n possible approximations. Among the stable realizations thus obtained, the one yielding the largest average inband noise attenuation, while respecting the chosen bound on the NTF-magnitude, is retained, if any exists. Note that the NTF-magnitude bound may have to be relaxed to a slightly larger value than that used for the original NTF design.

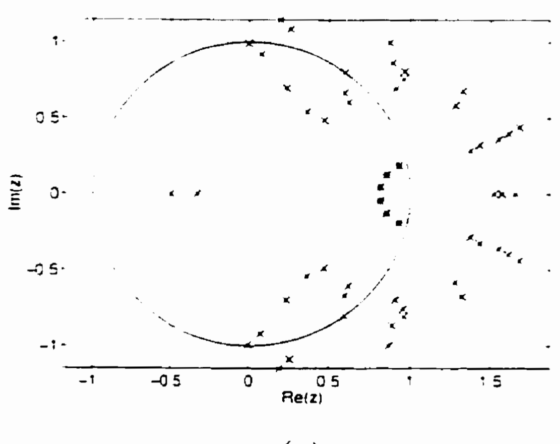
If a finer quantization is required, the coefficients can be realized by canonical signed digits (CSD), that is sums or differences of a few powers-of-two. This requires more hardware but results in smaller coefficient changes, which helps obtain a 'quantized' NTF closer to the desired one. It is a trade-off which the designer must address on a case-per-case basis. Note that the biquad coefficient in a bandpass modulator may have to be quantized to a CSD of as many as four terms so as to locate the signal band with enough precision.

This quantization process is simple in principle but even more computationally intensive than solving for the coefficients of a structure, and therefore the use of the CAD tool presented in Chapter 4, or a similar one, is absolutely essential here.

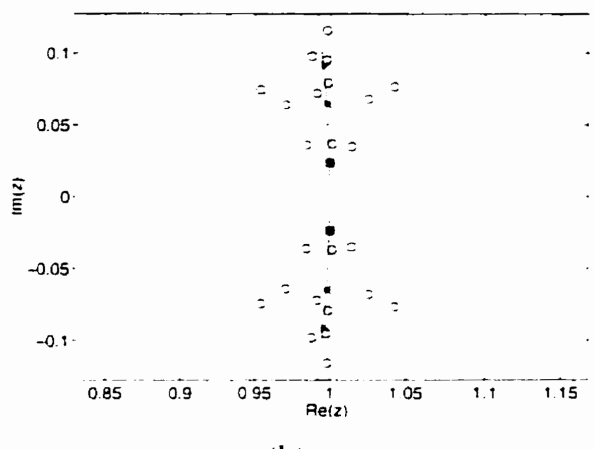
As a result of this quantization process, the zeros and poles of the NTF move off their initial locations, in a manner depending on the modulator structure's sensitivity to its coefficients. Fig. 2.17 shows the zeros and poles obtained by quantizing the coefficients for both our 6th order example NTF's to powers-of-two, for the integrator cascade structure. The initial poles and zeros, the ones of the modulator with non-powers-of-two coefficients, are marked by asterisks (*). The various possible locations of the poles when each coefficient is quantized either upward or downward are identified by the 'x' symbols on Figs. 2.17(a) and 2.17(c). Note that for clarity not all 2^{12} possible poles are shown but rather a representative subset of them. Similarly, the zeros of the possible realizations with quantized coefficients are represented with 'o' markers, on Figs. 2.17(b) and 2.17(d).

Our results show that the integrator cascade has serious drawbacks with respect to coefficient quantization. First, in many instances the zeros move far off the unit circle, decreasing the inband noise attenuation. Thus, designs realized with the integrator cascade with powers-of-two coefficients are likely to have a significantly poorer performance than is desired.

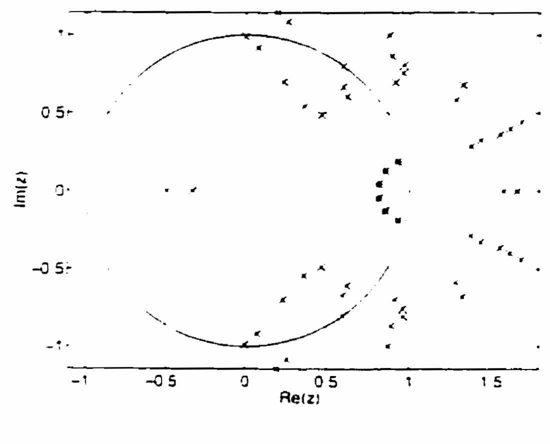
Much worse is the behavior of the NTF poles under quantization of the coefficients. For high-order, small-bandwidth modulators, all of the 2^n sets of quantized coefficients result either in poles lying outside the unit circle, or in a NTF magnitude exceeding the allowed maximum of 2.0 given in Section 2.5.1. In either case the resulting modulator



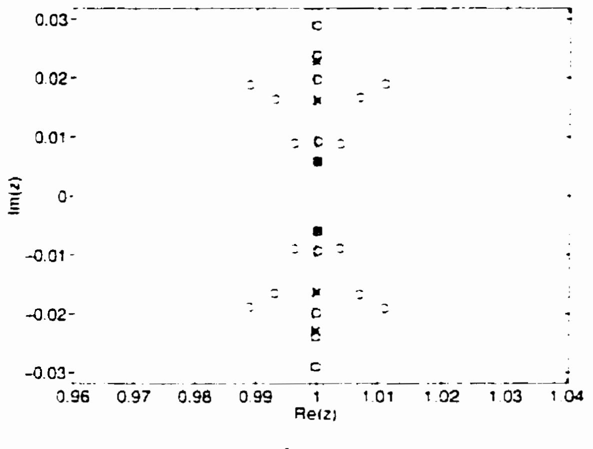
(a)



(b)



(c)

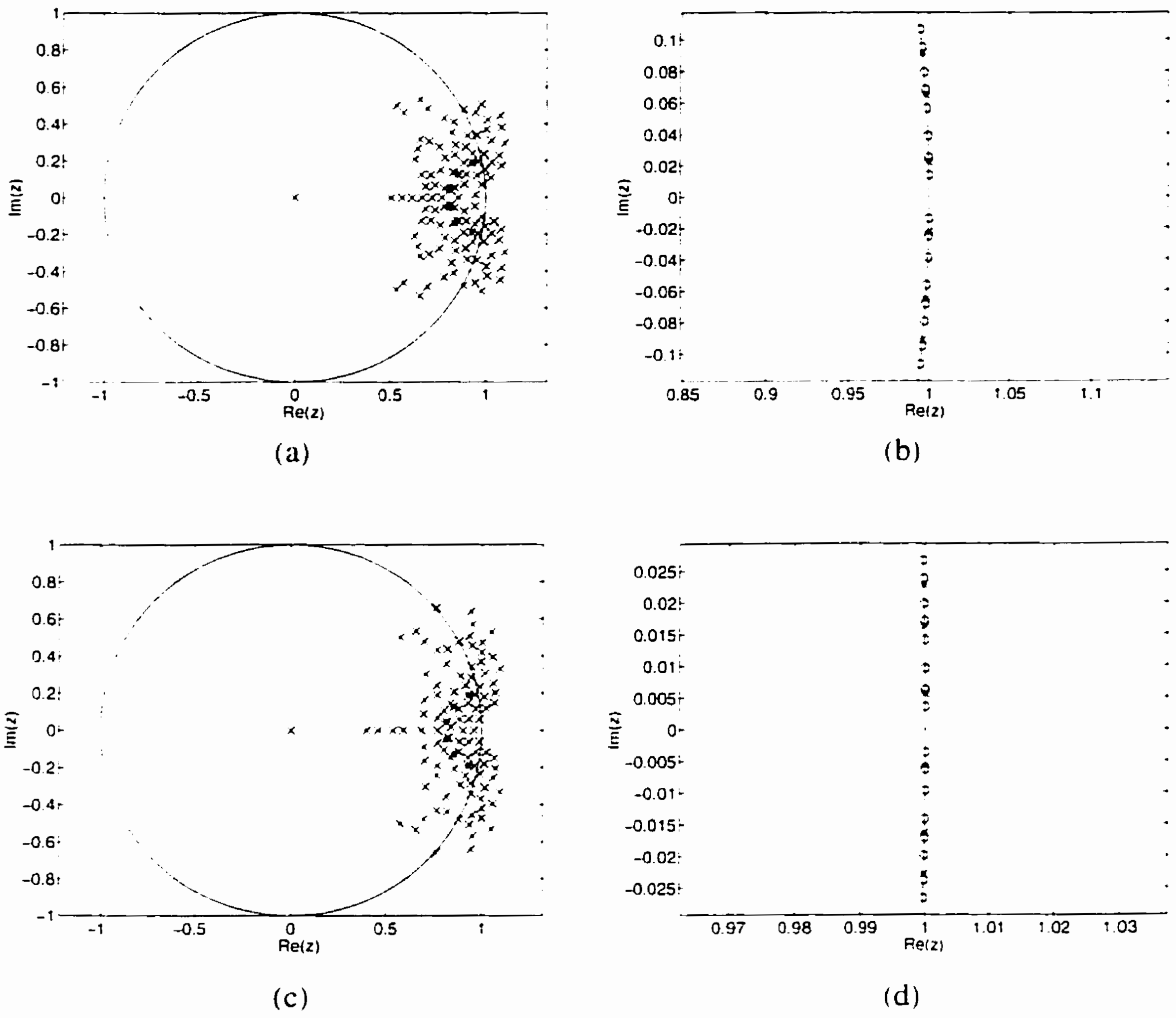


(d)

*: Poles/zeros before quantization
x: Possible poles after coefficient quantization
o: Possible zeros after coefficient quantization

Fig. 2.17: NTF zeros and poles realized by quantized coefficients in the integrator cascade modulator structure, for our 6th-order examples: (a) poles, OSR=32; (b) zeros, OSR=32; (c) poles, OSR=128; (d) zeros, OSR=128.

is unstable. Thus our design strategy fails completely for this structure and we conclude that it is extremely difficult to design high-order delta-sigma modulators with powers-of-two coefficients for the integrator cascade structure.



*: Poles/zeros before quantization
x: Possible poles after coefficient quantization
o: Possible zeros after coefficient quantization

Fig. 2.18: NTF zeros and poles realized by quantized coefficients in the LDI-ladder structure, for our 6th-order examples: (a) poles, OSR=32; (b) zeros, OSR=32; (c) poles, OSR=128; (d) zeros, OSR=128.

These results seem to indicate that coefficient quantization requires a realization having good coefficient sensitivity properties. We now turn to the effect of quantizing the coefficients of an LDI ladder realization. Just as Fig. 2.17, Fig. 2.18 shows the possible poles and zeros of the ladder-based structure with quantized coefficients. Note, in Figs. 2.18(b) and 2.18(d), that all the zeros are located on the unit circle, at angles close to their original values. Since no damping is present in the ladder, the poles of *H(z)*, which are also the zeros of the NTF, remain on the unit circle when the coefficients are quantized, which is not the case with the previous integrator cascade structure.

In addition, the NTF poles remain much closer to their original position than with the cascade structure (seen in Figs. 2.18(a) and 2.18(c)). This is explained by two facts. Considering that the poles of *H(z)* change very little when the B coefficients are quantized, the transfer functions from the input to each integrator output do not vary significantly either. Moreover, the change in a specific coefficient A_i can be partially cancelled by an opposite change in the corresponding feedforward coefficient B_i, thus keeping the transfer functions from each integrator output to the filter output nearly unchanged.

Overall, with powers-of-two coefficients only, the ladder structure is capable of realizing a close approximation to the desired modulator functionality described by an NTF pole-zero distribution such as those in Fig. 2.8 and Fig. 2.9. The quantized coefficients of both design examples are given in Table 2.1:

Table 2.1: Quantized coefficients of the 6th-order designs realized with an LDI-ladder topology, for OSR=32 and OSR=128

N=6, OSR=32		N=6, OSR=128	
LADDER FEEDBACK COEFFICIENTS	LADDER FEEDFORWARD COEFFICIENTS	LADDER FEEDBACK COEFFICIENTS	LADDER FEEDFORWARD COEFFICIENTS
$A_1 = 2^{-9}$	$B_1 = 2^{-1}$	$A_1 = 2^{-13}$	$B_1 = 2^{-1}$
$A_2 = 2^{0}$	$B_2 = 2^{7}$	$A_2 = 2^{-1}$	$B_2 = 2^{11}$
$A_3 = 2^{-8}$	$B_3 = 2^{4}$	$A_3 = 2^{-12}$	$B_3 = 2^{9}$
$A_4 = 2^{-1}$	$B_4 = 2^{10}$	$A_4 = 2^{-1}$	$B_4 = 2^{19}$
$A_5 = 2^{-8}$	$B_5 = 2^{6}$	$A_5 = 2^{-12}$	$B_5 = 2^{15}$
$A_6 = 2^{0}$	$B_6 = 2^{10}$	$A_6 = 2^{0}$	$B_6 = 2^{24}$

Fig. 2.19 displays the inband magnitude of the desired NTFs and of the ones realized by a multiplier-free ladder with the best set of quantized coefficients, for both 6th-order designs. The solid curves represent the NTF of the modulators with quantized coef-

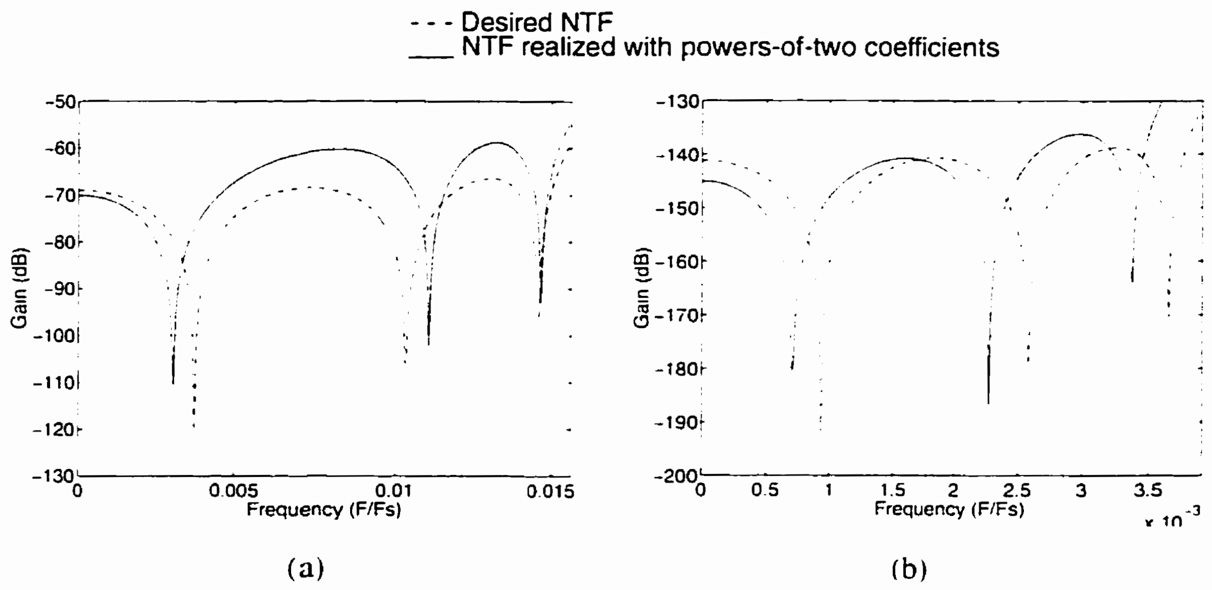


Fig. 2.19: Inband magnitude of the desired NTF (dashed line) and its approximation with powers-of-two ladder coefficients (solid line), for our 6th-order examples: (a) OSR=32, (b) OSR=128.

ficients, while the dashed curves depict the desired NTF. The difference between the two NTF's is at most 10 dB, both for OSR=32 and for OSR=128. These plots thus demonstrate that quantizing the coefficients of an LDI-ladder-based modulator is a viable way to reduce the hardware complexity without sacrificing too much performance. An Nth-order design with quantized coefficients requires $2N$ adders, N subtractors, $2N$ shift units, N registers and a sign detector.

As for the resonator cascade realization, it does not yield stable modulators with coefficients all equal to a single power-of-two, at least for our two NTF examples. However, if the B coefficients are quantized to sums or differences of 2 powers-of-two, i.e. 2-term CSDs, a good approximation of the desired NTF is obtained.

For bandpass modulators, the biquad coefficient C must also be quantized, generally to more than a single term so as to locate the signal band of the quantized design with enough precision.

2.5.5 Simulation

The modulator design cycle is not complete until simulations and prototypes have demonstrated the stability of the design for the intended inputs. There are two reasons for this. First, a delta-sigma modulator is a system with a hard non-linearity and thus its stability is conditional on the input level in a way which is difficult to predict. Second, in a digital implementation, finite-register length effects may create noise at significant levels, or even instability.

Simulation of the delta-sigma modulator, assuming floating-point precision, is easily performed by a simple C program. A given modulator design can be simulated at various signal levels so as to determine the maximum SNR and the corresponding input level, as well as the available dynamic range. This was done for both our 6th order examples. The inputs were sinusoids having periods 128 and 512, respectively, for the designs with OSR=32 and OSR=128. (The periods are powers-of-2 to ensure coherence of the input with the observation interval of 2^{14} samples). The input amplitude was varied from 10^{-4} to 0.9 so as to capture the input level at which the modulator becomes unstable (i.e. when the SNR suddenly drops). The output was then processed using the continuous-fifth-derivative window found in [45]. This window has rapidly decaying sidelobes in the frequency domain and thus dramatically reduces the frequency smearing due to the unavoidable incoherence of the delta-sigma modulator output, as explained in Appendix A.

Fig. 2.20 shows the results of these amplitude sweep simulations for both designs. One can clearly notice that the SNR suddenly drops at input levels close to 0.5. This happens because the modulator is no longer stable. For the modulator with an OSR of 32, the available dynamic range is 90 dB. For the modulator with an OSR of 128, it is 160 dB.

Fig. 2.21 shows the inband power spectral density of the simulation outputs for a 0.5 signal amplitude. Clearly, the noise power assumes the spectral distribution predicted by the NTF in Fig. 2.12, assuming a white quantization noise (the predicted noise spectrum is shown as a dashed curve). The zeros can be seen at the predicted frequencies and the signal shows no harmonic distortion. The output of the modulator with OSR=32 displays an SNR of 80 dB, while the one with OSR=128 has an SNR of 155 dB.

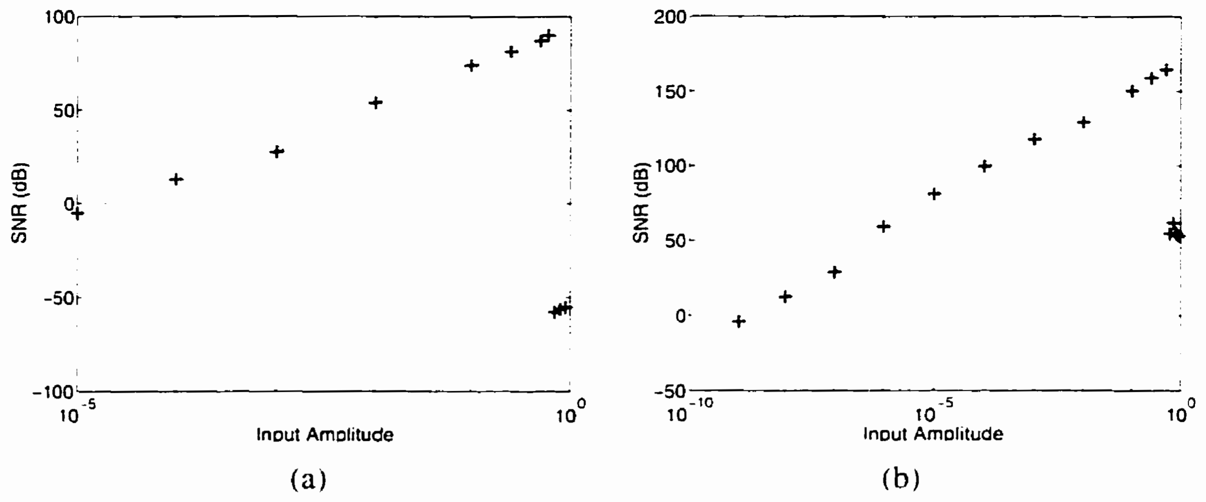


Fig. 2.20: Simulated SNR versus input amplitude for the 6th-order designs: (a) OSR=32, (b) OSR=128.

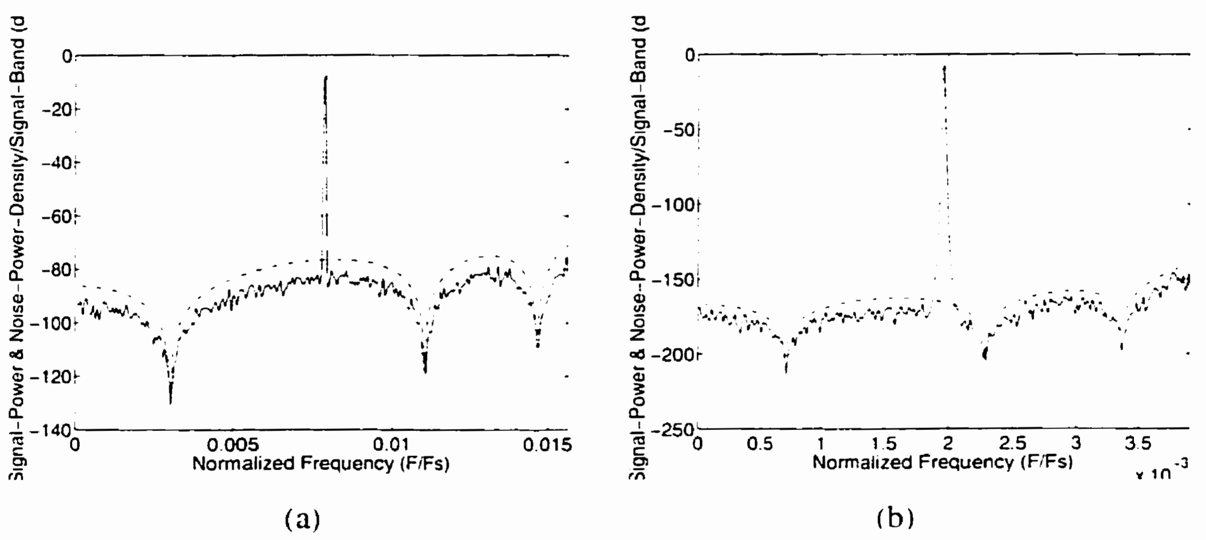
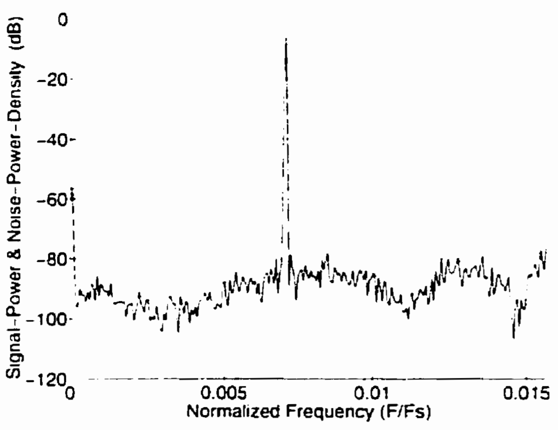


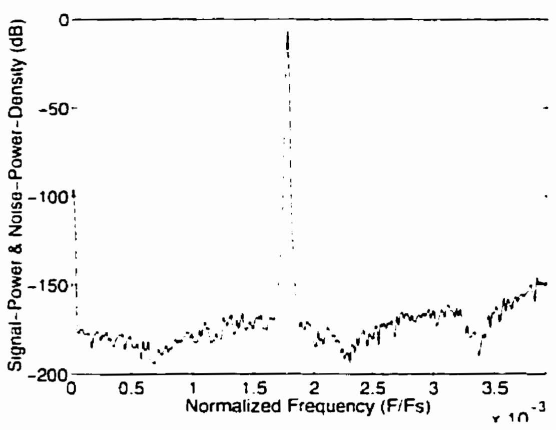
Fig. 2.21: Simulated inband power spectrum for the 6th-order designs: (a) OSR=32, (b) OSR=128.

2.5.6 Prototyping

Prototyping can be accomplished by synthesizing an HDL description of the circuit to a Field-Programmable-Gate-Array (FPGA). The register lengths can be modified until a prototype using minimum hardware and yielding the expected SNR performance is



(a)



(b)

Fig. 2.22: inband power spectra for the 6th-order prototypes: (a) OSR=32, (b) OSR=128.

obtained. The CAD tool described in Chapter 4 can generate VHDL code describing a given modulator design. Such code has been compiled to implement the 6th-order modulator with OSR=32 onto a Xilinx 4010 FPGA with a 16-bit register length and 3 integer bits. The inband spectral density of its output when stimulated by a sinusoidal signal of amplitude 0.5 and period 142.2 is shown in Fig. 2.22(a). Likewise, Fig. 2.22(b) displays the inband spectrum for the OSR=128 prototype with an input sinusoid of amplitude 0.5 and period 568.9, which was implemented using a register length of 32 bits and 3 integer bits. The spectra were obtained by sampling the digital bit stream at the output of the FPGA and performing a Fast Fourier Transform on the data. Notice that the noise spectra are virtually identical to the ones obtained from floating-point simulations, shown in Fig. 2.21. The major difference is that the zeros of the NTF cannot be seen as sharply defined for the prototypes as for the simulations. This is due to the noise floor generated by the finite register lengths used in the prototypes. A DC error is also seen on these plots at a level significantly higher than the noise floor. The source of this offset is unknown, and it was not seen in the results of simulations performed on the VHDL description of the prototypes. This DC component was ignored when computing the SNR.

The SNR figures are comparable to those obtained from simulations, namely 80 dB and 155 dB, respectively for the low- and the high-OSR design. The prototypes thus validate the assumption that the fixed-point implementations of the modulator designs behave as predicted by the linear model and by simulation. Bandpass and highpass prototypes can be realized and tested in the same fashion with similar results.

2.5.7 Optimal NTF Design

An empirical study of one-bit delta-sigma modulators [25] gives the maximum SNR that can be achieved by a modulator of a given order and OSR. The parameter which must be optimized for maximum SNR is the NTF bound. The CAD tool presented in Chapter 4 is capable of finding the optimal NTF bound.

The method consists of gradually increasing the NTF bound, and for each value thereof, simulating the modulator with a variety of tone amplitudes and recording the maximum SNR achieved over all these amplitudes. While it is a brute-force method, the use of the CAD tool makes it reasonably fast, as described in more detail in Section 4.4.7.

2.6 Conclusion

This chapter contains two important contributions. The first is an overall modulator topology for which the STF is unity, and this is essential for use in delta-sigma oscillator. The second is a complete design framework and internal modulator topology, the LDI-ladder realization. Together, this method and topology yield hardware-efficient realizations of high-order digital one-bit delta-sigma modulator, whether for lowpass, highpass, or arbitrary bandpass signal bands, based on quantizing the modulator coefficients to powers-of-two. Finally, simulations and prototypes prove the validity and practicality of these modulator designs.

Chapter 3 Arbitrary-Precision Delta-Sigma Oscillators

3.1 Introduction

As pointed out in Chapter 1, a general delta-sigma oscillator topology is introduced and explained in detail in this chapter. It overcomes the limited stability of Lu, Roberts and Johns' [5] original 2nd-order design (shown in Fig. 1.2), offers complete flexibility in setting the location of the signal band in the Nyquist interval, and allows for the use of a high-order modulator, for the purpose of reducing the inband noise power.

The generalized delta-sigma oscillator topology is shown in Fig. 3.1. Like Lu's design, it is constructed from a pair of discrete time integrators, a delta-sigma modulator, and a multiplexer which implements the loop-gain coefficient k_0. It also displays two new features. First of all, the delta-sigma modulator is not any more the traditional 2nd-order circuit with two zeros at DC, but a modulator of arbitrary order and topology whose Signal Transfer-Function (STF) is equal to 1. The second new feature is an additional feedback loop around the integrators which bypasses the modulator and multiplexer and implements a scaling coefficient k_1.

The special requirement of unity-STF imposed on the modulator is justified in Section 3.2, which presents a linear model for the oscillator. Section 3.3 shows how the stability of the circuit depends on the scaling coefficient in the modulator-multiplexer loop, k_0.

while Section 3.4 demonstrates how the extra loop coefficient k_1 is used to keep the oscillator stable by maintaining the multiplexed coefficient k_0 under an acceptable threshold, no matter what the generated tone frequency is. It is also shown how to implement k_1 with a minimal amount of hardware.

This chapter goes beyond the principles of operation and the design of single-tone, lowpass delta-sigma oscillators. Section 3.5.1 explains how the same generalized topology can be used to generate signals over any arbitrary band within the Nyquist interval, i.e. to realize bandpass and highpass oscillators. Some simple modifications to the hardware, described in Section 3.5.2, result in multitone generators, useful in particular to excite a circuit for intermodulation distortion measurements. Section 3.5.3 explains how the noise-spectrum of the generated signal can be optimized in the context of mixed-signal self-test.

3.2 Linear Model

As was explained in Section 2.3, the delta-sigma modulator embedded in the oscillator as shown in Fig. 3.1 can be represented by a linear model, even though it is a non-linear circuit. This model consists of a Signal-Transfer-Function (STF) from input to output.

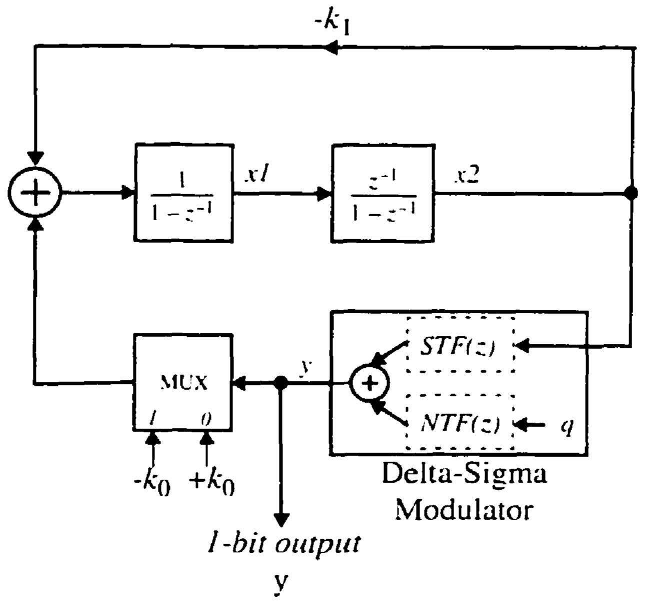


Fig. 3.1: Generalized Delta-Sigma Oscillator (DSO).

and a Noise-Transfer-Function (NTF) from the quantization noise input q, internal to the modulator, to its output. Typically, the quantization error q introduced by the one-bit quantizer inside a delta-sigma modulator is assumed to be additive white noise (this was discussed in detail in Section 2.3). The modulator is then characterized by $STF(z)$ and $NTF(z)$, according to the following equations:

$$Y(z) = STF(z)X_2(z) + NTF(z)Q(z), \tag{3.1}$$

where $Y(z)$, $X_2(z)$ and $Q(z)$ are the Z-transforms of $y(n)$, $x_2(n)$ and $q(n)$, respectively.

The linear model of the delta-sigma modulator, summed up by Eqn. (3.1), implies a similar model for the delta-sigma oscillator circuit. The first part of this model ignores the quantization noise and predicts the oscillatory behavior of the circuit, and is the counterpart to the STF modelling of a modulator's behavior with respect to its input signal. The second part of the model consists of a new noise-transfer-function $NTF'(z)$, which, along with an estimate of the Power Spectral Density (PSD) of $Q(z)$, describes the oscillator's output noise spectrum.

In order to derive the signal behavior of the oscillator, let us assume for now that the noise input q of the modulator in Fig. 3.1 is zero, or equivalently set $Q(z)$ to 0 in Eqn. (3.1). The circuit resonates if its loop gain has the following form:

$$L(z) = \frac{kz^{-1}}{(1-z^{-1})^2}. \tag{3.2}$$

In other words, the characteristic polynomial of the linear system must be:

$$1 + (k-2)z^{-1} + z^{-2}, \tag{3.3}$$

where k is the total loop gain in the system. In that case the solution to the difference equations describing the output of the system with no input will be a sinusoid depending on the initial conditions ($x_1(0)$ and $x_2(0)$) and the loop gain k as described by Eqn. (1.1), provided that $0<k<4$.

The actual loop gain of the circuit of Fig. 3.1 is found by breaking the loop at the output of either of the integrators and is given by:

$$L(z) = \frac{(k_0 STF(z) + k_1)z^{-1}}{(1-z^{-1})^2}. \tag{3.4}$$

Comparing Eqn. (3.2) with Eqn. (3.4) indicates that the circuit will resonate, i.e. produce a sinusoidal output, provided that *STF(z)* is unity for all *z*. In that case the total loop gain is given by:

$$k = k_0 + k_1 \tag{3.5}$$

and the frequency of oscillation, amplitude and phase of the output tone are still given by Eqns. (1.2), (1.3) and (1.4), respectively.

Let us now develop an expression for the noise spectrum created by the oscillator circuit, due to the quantization noise in the delta-sigma modulator. The oscillator can be considered as a linear filter with an input q located within the modulator. For purposes of computing the inband noise, this input is considered to be white noise, even though in fact it is correlated with the signal being modulated. If *NTF(z)* is the transfer function from q to the modulator output, then we call *NTF'(z)* the oscillator's Noise-Transfer-Function, that is the transfer function from q to the oscillator output y. It is given by:

$$NTF'(z) = \frac{Y(z)}{Q(z)} = \frac{1 + (k_1 - 2) z^{-1} + z^{-2}}{1 + (k_0 + k_1 - 2) z^{-1} + z^{-2}} \cdot NTF(z) \tag{3.6}$$

In addition to the poles and zeros of the modulator NTF, *NTF'(z)* has a pair of zeros and a pair of poles located on the unit circle in the Z-plane (i.e. at physical frequencies), provided that $0 < |k_1| < 4$ and $0 < |k_0 + k_1| < 4$. The new zeros have a radial position equal to $\pm \mathrm{acos}(1 - |k_1|/2)$ while the new poles are located at $\pm \mathrm{acos}(1 - |k_0 + k_1|/2)$, the frequency of the generated sinusoid. The presence of the added zeros, if they fall in the signal band, is somewhat beneficial in reducing the total amount of inband noise power. The poles on the unit circle represent the fact that the system is critically stable.

In Section 2.3 an estimate of the PSD of *Q(z)* was derived for delta-sigma modulators and given by Eqn. (2.9). The same argument leads to an equivalent expression for the PSD of the quantization noise in the oscillator case:

$$S_Q = \frac{1 - P_X}{\int_0^1 |NTF'(e^{j2\pi f})|^2 df}. \tag{3.7}$$

Note that $NTF'(z)$ has taken the place of $NTF(z)$ in Eqn. (2.9). What remains to be determined to complete the model is the signal power at the output of the modulator, P_S. Note that it is equal to the power of the modulator input signal x_2 since $STF(z)=1$.

According to the previous discussion, we assume x_2 to be a sinusoidal signal and will prove the validity of this assumption later. If the amplitude is programmed to be equal to A, then the PSD of the noise component of the output, according to Eqn. (3.7), is given by

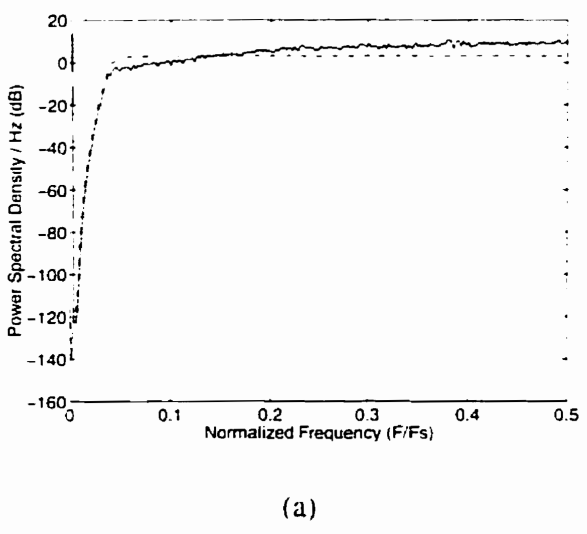
$$\frac{\left(1-\frac{A^2}{2}\right)\left|NTF'(e^{j2\pi f})\right|^2}{\int_0^1 \left|NTF'(e^{j2\pi f})\right|^2 d\omega}. \tag{3.8}$$

The signal y feeds back toward the input of the modulator, seeing a transfer function given by:

$$\frac{X_2(z)}{Y(z)} = \frac{k_0 z^{-1}}{1+(k_1-2)z^{-1}+z^{-2}}. \tag{3.9}$$

If k_0 is small, then this transfer function is small everywhere except close to the pole frequency and we can assume that the out-of-band noise is sufficiently attenuated for x_2 to be effectively modelled by a sinusoid, thereby validating our initial assumption.

A 6th-order design has been simulated to prove the validity of Eqn. (3.8). The modulator is the same as the one used in the example of Section 2.3. The tone period is set to 1024, or equivalently the frequency is 0.00098, implemented with k_0=-2.34x10^{-5} and k_1=2^{-14}=6.10x10^{-5}. Fig. 3.2(a) shows the spectral content of the output signal over the Nyquist interval for a signal amplitude equal to 0.25 (-12 dB). Fig. 3.2(b) shows the inband power spectrum (represented by the solid line), as well as the spectrum predicted by the linear model (shown with a dotted line). Note that the continuous fifth-derivative window has been applied to the data before the FFT was computed, as explained in Appendix A. The SNR of the generated tone over the displayed frequency band is 130 dB. This is 60 dB better than the 70 dB achieved by the 2nd order oscillator over the same signal bandwidth (Fig. 1.4). This example illustrates the gain in signal precision obtained when a higher-order modulator replaces the 2nd-order modulator, and the possibility to



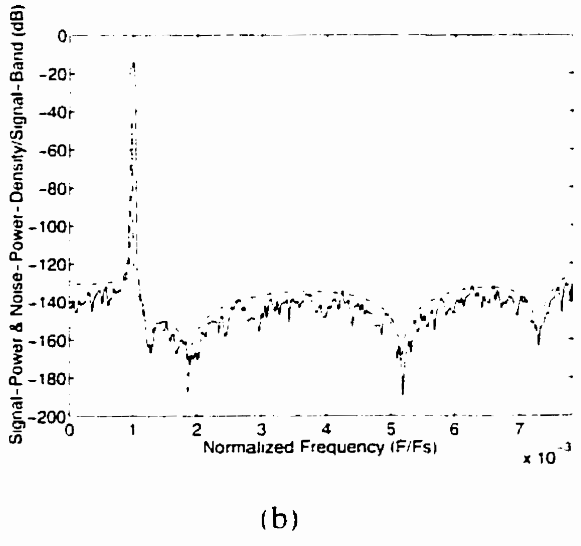


Fig. 3.2: (a) Nyquist-band and (b) Inband power spectrum for a 6th-order oscillator. The solid line represents simulation data, while the dotted line represents the noise spectrum predicted by the linear model.

achieve virtually limitless SNR (for all practical purposes) over small signal bandwidths, as set out in Section 1.3.3.

3.3 Stability Study

3.3.1 Classification of Long-Term Behavior

Just like delta-sigma modulators, delta-sigma oscillators are non-linear systems, and thus their stability properties depend on the input signal and cannot be predicted by frequency domain methods. However, the fact that the linearized system described by Eqn. (3.4) (with no noise input) is critically stable gives us hope that the actual non-linear system may, in some cases, be critically stable too. Unfortunately it is not known how to prove it rigorously. The stability of delta-sigma modulators is in general equally difficult to assess rigorously, although in their case empirical findings [25] compensate for our lack of theoretical knowledge. What is thus needed here is the equivalent empirical study applying to delta-sigma oscillators. This section presents such a study.

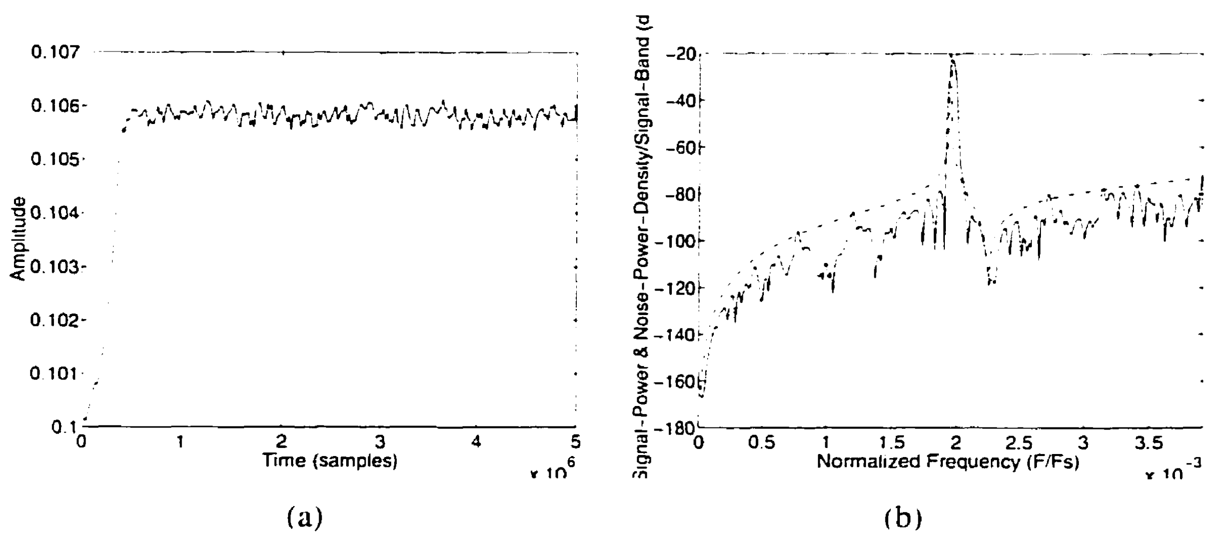


Fig. 3.3: Simulation of a 2nd-order oscillator: (a) tone amplitude; (b) power spectral density of a 64k sample taken after 10^8 iterations.

Some non-linear effects can still be predicted in a qualitative way, using the linear model. Because the delta-sigma modulator output can be equal to +1 or -1 only, the state variables of the oscillator (x_1 and x_2) loop can only take on a set of quantized values. After each period of oscillation, these state variables may have values which correspond to a slightly different amplitude of oscillation. This imprecision will affect the amplitude of oscillation over time. We can foresee two cases, one in which the amplitude of oscillation converges to a value, hopefully close to the intended amplitude, and another one in which the amplitude drifts away without bound, until the modulator reaches instability.

Fig. 3.3(a) illustrates the first of these possibilities. The amplitude is initially set to 0.1, but settles to 0.106 after about 500,000 samples. The simulation shown here lasted 5,000,000 iterations, after which time an FFT of the output was computed. It is shown in Fig. 3.3(b). The tone displays an SNR of 68 dB.

The other case, in which the tone amplitude changes slowly but with no bound, is illustrated by Fig. 3.4(a), which shows the simulated amplitude of a 6th order oscillator with OSR 64. The period is 256. The tone amplitude is initially set to 0.1 but has decreased to 0.03 after 10^8 samples. However it changes so slowly that at any given time the quality of the signal is as predicted by the linear model. Fig. 3.4(b) shows the FFT of

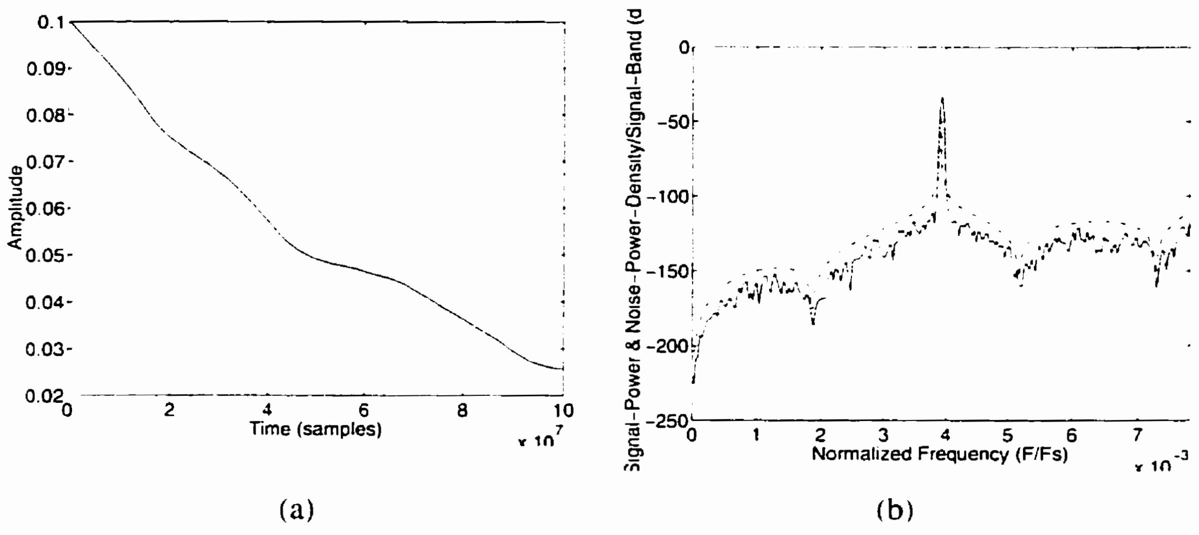


Fig. 3.4: Simulation of a 6th-order oscillator: (a) tone amplitude; (b) power spectral density of a 64k sample taken after 10^8 iterations.

the simulated output over a 64k interval, taken at the end of the simulation, after 10^8 samples. Although the signal power is -30 dB rather than the intended -20 dB, the SNR is 96 dB. This oscillator could still be very useful in an application which can compensate for the variation in tone amplitude.

In the spectral domain, these departures from the linear behavior can be said to be caused by the noise generated by the modulator. This noise is scaled by k_0 before being injected into the resonator. If k_0 is large, then the disturbance created by this noise is likely to throw the oscillator into instability very rapidly. *If, on the contrary, k_0 is small, then it is reasonable to assume that the resulting closed-loop circuit may display the same oscillatory behavior as the pure digital resonator* (as argued in the previous section), at least over some significant period of time after the initial conditions have been set. In other words if the noise injected in the resonator loop is small enough, we expect the oscillator to be able to compensate for this disturbance. For this reason, k_0 must be kept under a certain threshold to prevent the injected noise from throwing the system into instability. The object of our empirical study is thus to find this threshold for various oscillator designs.

3.3.2 Method

In order to obtain an overall estimate of what constitutes an appropriately small value of k_0, extensive simulations were conducted, for a variety of lowpass oscillator designs. Six designs were simulated, using modulators of orders 2, 4, and 6, and for OSRs equal to 32 and 128. In each case the modulator was designed to offer the maximum SNR for the given order and OSR, as described in Section 2.5.7, using the resonator cascade structure with non-quantized coefficients (note that the modulator coefficients need not be quantized for this study, and thus any modulator structure will yield the same results). The oscillators were designed with $k_1=0$, i.e. with no feedback loop other than through the multiplexer coefficient k_0, as in the original 2nd-order delta-sigma oscillator of [5].

Increasing values of k_0 were used; they were chosen so as to make visible the maximum range for which the various oscillator designs are stable, with the tone always being generated within the signal band. Increasing tone amplitudes were also used. In each case the time during which the oscillator remained stable was recorded, up to a maximum of 10^8 samples. This is an upper bound on the requirements of a practical test, i.e. a 1 second test at a 100 MHz rate. Stability was defined as follows: the modulator is stable as long as the modulator quantizer input is less than ten times larger than the quantizer output level, in absolute value. This criterion is arbitrary but, from experience, captures the essential facts about modulator and oscillator stability. When the oscillator was still stable at the end of the simulation, the amplitude and SNR of the generated tone were computed.

3.3.3 Results

Note that only results for lowpass oscillators are presented here. Similar results hold for bandpass and highpass oscillators.

The following six tables contain the simulation results. Each column corresponds to a given value of k_0. The value of the tone frequency is given relative to the sampling rate as f, and relative to the upper signal-band-edge frequency as f/f_u. A_i is the tone amplitude programmed by the initial register values in the oscillator. Each table cell is shaded so as to represent the outcome of the simulation in the following manner. If at the end of the simulation the tone amplitude was still within 10% of the intended value, the cell is not shaded and displays the final tone amplitude A_f and the SNR at the end of the simulation. If the modulator was still stable but the tone amplitude had departed from the intended value by more than 10%, the corresponding table cell is lightly shaded, and again contains the values of A_f and of the SNR. Finally, when the modulator had become unstable before the end of the simulation time, the cell is a darker shade of grey and displays the simulation iteration at which instability was reached.

Table 3.1: Oscillator Stability Results for N=2, OSR = 32

k_0	$k_0=10^{-6}$	$k_0=10^{-5}$
f	$1.6x10^{-4}$	$5.0x10^{-4}$
f/fu	0.010	0.032
A_i = 0.025	A_f = 0.0467 SNR = 36.9 dB	A_f = 0.264 SNR = 51.7 dB
A_i = 0.1	A_f = 0.109 SNR = 45.6 dB	A_f = 0.487 SNR = 56.4 dB
A_i = 0.25	A_f = 0.344 SNR = 54.1 dB	A_f = 0.796 SNR = 56.0 dB
A_i = 0.5	A_f = 0.542 SNR = 57.8 dB	A_f = 0.808 SNR = 55.5 dB
A_i = 0.75	A_f = 0.780 SNR = 56.7 dB	A_f = 0.781 SNR = 56.5 dB

Table 3.2: Oscillator Stability Results for N=2, OSR=128

k_0	10^{-6}	10^{-5}	10^{-4}	$5x10^{-4}$
f	$1.6x10^{-4}$	$5.0x10^{-4}$	$1.6x10^{-3}$	$3.6x10^{-3}$
f/fu	0.041	0.129	0.407	0.911
A_i = 0.025	A_f = 0.0245 SNR = 60.7 dB	A_f = 0.0241 SNR = 60.9dB	A_f = 0.0258 SNR = 58.1 dB	A_f = 0.0248 SNR = 53.2 dB
A_i = 0.1	A_f = 0.0997 SNR = 74.9 dB	A_f = 0.100 SNR = 74.5 dB	A_f = 0.0960 SNR = 71.8 dB	A_f = 0.0923 SNR = 65.3 dB

Table 3.2: Oscillator Stability Results for N=2, OSR=128

k_0	10^{-6}	10^{-5}	10^{-4}	$5x10^{-4}$
f	$1.6x10^{-4}$	$5.0x10^{-4}$	$1.6x10^{-3}$	$3.6x10^{-3}$
f/fu	0.041	0.129	0.407	0.911
A_i = 0.25	A_f = 0.250 SNR = 82.4 dB	A_f = 0.249 SNR = 81.0 dB	A_f = 0.246 SNR = 79.8 dB	A_f = 0.243 SNR = 72.3 dB
A_i = 0.5	A_f = 0.503 SNR = 87.2 dB	A_f = 0.504 SNR = 87.7 dB	A_f = 0.517 SNR = 86.3 dB	A_f = 0.301 SNR = 73.1 dB
A_i = 0.75	A_f = 0.752 SNR = 83.1 dB	A_f = 0.751 SNR = 86.2 dB	A_f = 0.752 SNR = 89.3 dB	A_f = 0.744 SNR = 80.2 dB

Table 3.3: Oscillator Stability Results for N=4, OSR=32

k_0	10^{-6}	10^{-5}	10^{-4}
f	$1.6x10^{-4}$	$5.0x10^{-4}$	$1.6x10^{-3}$
f/fu	0.010	0.032	0.102
A_i = 0.025	A_f = 0.0247 SNR = 54.4 dB	A_f = 0.0247 SNR = 54.1 dB	A_f = 0.0234 SNR = 52.9 dB
A_i = 0.1	A_f = 0.0997 SNR = 66.3 dB	A_f = 0.104 SNR = 65.7 dB	unstable at t=2.320e+07
A_i = 0.25	A_f = 0.274 SNR = 74.0 dB	unstable at t=9.374e+07	unstable at t=1.045e+07
A_i = 0.5	A_f = 0.535 SNR = 79.7 dB	unstable at t=1.311e+07	unstable at t=3.074e+06
A_i = 0.75	unstable at t=181	unstable at t=930	unstable at t=318

Table 3.4: Oscillator Stability Results for N=4, OSR=128

k_0	10^{-6}	10^{-5}	10^{-4}	$5x10^{-4}$
f	$1.6x10^{-4}$	$5.0x10^{-4}$	$1.6x10^{-3}$	$3.6x10^{-3}$
f/fu	0.041	0.129	0.407	0.911
A_i = 0.025	A_f = 0.0248 SNR = 108.9 dB	A_f = 0.0246 SNR = 108.0 dB	A_f = 0.0256 SNR = 104.2 dB	A_f = 0.0298 SNR = 96.8 dB

Table 3.4: Oscillator Stability Results for N=4, OSR=128

k_0	10^{-6}	10^{-5}	10^{-4}	$5x10^{-4}$
f	$1.6x10^{-4}$	$5.0x10^{-4}$	$1.6x10^{-3}$	$3.6x10^{-3}$
f/f_u	0.041	0.129	0.407	0.911
A_i = 0.1	A_f = 0.100 SNR = 119.0 dB	A_f = 0.101 SNR = 119.1 dB	A_f = 0.996 SNR = 113.7 dB	A_f = 0.113 SNR = 110.4 dB
A_i = 0.25	A_f = 0.250 SNR = 127.2 dB	A_f = 0.251 SNR = 127.1 dB	A_f = 0.247 SNR = 117.4 dB	A_f = 0.286 SNR = 116.0 dB
A_i = 0.5	A_f = 0.500 SNR = 132.3 dB	A_f = 0.501 SNR = 133.3 dB	unstable at t=5.615e+07	unstable at t=8.308e+07
A_i = 0.75	unstable at t=4.900e+01	unstable at t=4.600e+01	unstable at t=4.500e+01	unstable at t=1.760e+02

Table 3.5: Oscillator Stability Results for N=6, OSR=32

k_0	10^{-6}	$1x10^{-5}$	$1x10^{-4}$
f	$1.6x10^{-4}$	$5.0x10^{-4}$	$1.6x10^{-3}$
f/f_u	0.010	0.032	0.102
A_i = 0.025	A_f = 0.0248 SNR = 68.4 dB	A_f = 0.00248 SNR = 67.8 dB	A_f = 0.0289 SNR = 68.1 dB
A_i = 0.1	A_f = 0.104 SNR = 81.2 dB	A_f = 0.154 SNR = 83.0 dB	unstable at t=4.707e+07
A_i = 0.25	A_f = 0.261 SNR = 87.7 dB	A_f = 0.375 SNR = 90.9 dB	unstable at t=2.219e+07
A_i = 0.5	unstable at t=3.645e+05	unstable at t=8.456e+04	unstable at t=1.618e+05

Table 3.6: Oscillator Stability Results for N=6, OSR=128

k_0	10^{-6}	$1x10^{-5}$	$1x10^{-4}$	$5x10^{-4}$
f	$1.6x10^{-4}$	$5.0x10^{-4}$	$1.6x10^{-3}$	$3.6x10^{-3}$
f/f_u	0.041	0.129	0.407	0.911
A_i = 0.025	A_f = 0.0250 SNR = 146.6 dB	A_f = 0.025 SNR = 147.5 dB	A_f = 0.0250 SNR = 144.3 dB	A_f = 0.0251 SNR = 140.2 dB

Table 3.6: Oscillator Stability Results for N=6, OSR=128

k_0	10^{-6}	$1x10^{-5}$	$1x10^{-4}$	$5x10^{-4}$
f	$1.6x10^{-4}$	$5.0x10^{-4}$	$1.6x10^{-3}$	$3.6x10^{-3}$
f/fu	0.041	0.129	0.407	0.911
A_i = 0.1	A_f = 0.100 SNR = 157.1 dB	A_f = 0.100 SNR = 157.0 dB	A_f = 0.100 SNR = 154.5 dB	A_f = 0.100 SNR = 157.1 dB
A_i = 0.25	A_f = 0.250 SNR = 165.6 dB	A_f = 0.250 SNR = 165.2 dB	A_f = 0.250 SNR = 163.7 dB	A_f = 0.251 SNR = 165.0 dB
A_i = 0.5	unstable at t=9163	unstable at t=6989	unstable at t=1629	unstable at t=4712
A_i = 0.75	unstable at t=38	unstable at t=39	unstable at t=42	unstable at t=146

The following trends are clearly visible from the collected data. First of all, oscillators designed for the lower OSR value (i.e. 32) remain stable only for small values of k_0, such as 10^{-6}. When k_0 is set to 10^{-5} the low-OSR oscillators remain stable only for small signal amplitudes. This is true for all three orders in these results, namely 2, 4 and 6. Another trend is that as the modulator order is increased, the range of stable input amplitudes diminishes. This is not surprising, given that the same fact holds for delta-sigma modulators. Finally, one should note the accuracy of the tone amplitude for the 6th-order design with OSR=128, throughout the entire range of values of k_0. (These values of k_0 span most of the signal band for an OSR of 128). Very little variation in amplitude is observed.

These results, which were collected with k_1 set to 0, lead to the expected conclusion that k_0 must be kept smaller than some threshold value. This will be accomplished in the next section by letting k_1 take non-zero values.

3.4 Minimal Realization of Stable Oscillator

3.4.1 Design Method

We now turn to the problem of realizing an oscillator capable of generating stable tones over a given signal band. If the signal band extends from the radial frequencies ω_a to ω_b, then the total loop gain k must be programmable in the range $[k_a, k_b]$ given by:

$$k_{a,b} = 2(1 - \cos\omega_{a,b}) \qquad (3.10)$$

An empirical study similar to the ones presented in Section 3.3 must be conducted so as to obtain a bound on k_0 which ensures enough stability. Once this bound is known, a set of discrete values for k_1 are chosen, so that the difference between any two adjacent values of k_1 is less than twice the bound on k_0. These discrete values of k_1 are chosen so that they can all be realized by sums and differences of only a few powers-of-two. An algorithm which computes a set of appropriate values of k_1 and their CSD realizations is implemented in the CAD tool and explained in Section 4.4.5.

3.4.2 Design Example

As an example, consider the design used in the previous section for N=4 and OSR=128. The values of k for which the tone frequency spans the signal band are in the interval [0.0, 6.02×10^{-4}]. The stability study performed in Section 3.3 revealed that the oscillator is stable for signal amplitudes up to 0.5 for $k_0=10^{-5}$ but not for $k_0=10^{-4}$. We thus set the bound on k_0 to 5×10^{-5}. A possible set of CSD values for k_1 such that k_0+k_1 spans the interval [0.0,6.02×10^{-4}] with the constraint that $k_0<5\times10^{-5}$ is then given below in Table 3.7.

Table 3.7: Discrete values of k1 and their CSD realizations

k_1	CSD REALIZATION
3.05×10^{-5}	2^{-15}
1.22×10^{-4}	2^{-13}

Table 3.7: Discrete values of k1 and their CSD realizations

k_1	CSD REALIZATION
2.14×10^{-4}	$2^{-12} - 2^{-15}$
3.05×10^{-4}	$2^{-12} + 2^{-14}$
3.66×10^{-4}	$2^{-11} - 2^{-13}$
4.58×10^{-4}	$2^{-11} - 2^{-15}$
5.49×10^{-4}	$2^{-11} + 2^{-14}$
6.10×10^{-4}	$2^{-11} + 2^{-13}$

These values can be realized by the simple arrangement shown in Fig. 3.5, using five fixed-shift units, three multiplexers, one adder and one numeric inverter. The fixed- or hard-wired-shift units implement the multiplications by the powers-of-two 2^{-11} to 2^{-15}. To program the oscillator to operate at a given frequency, the following method applies. First

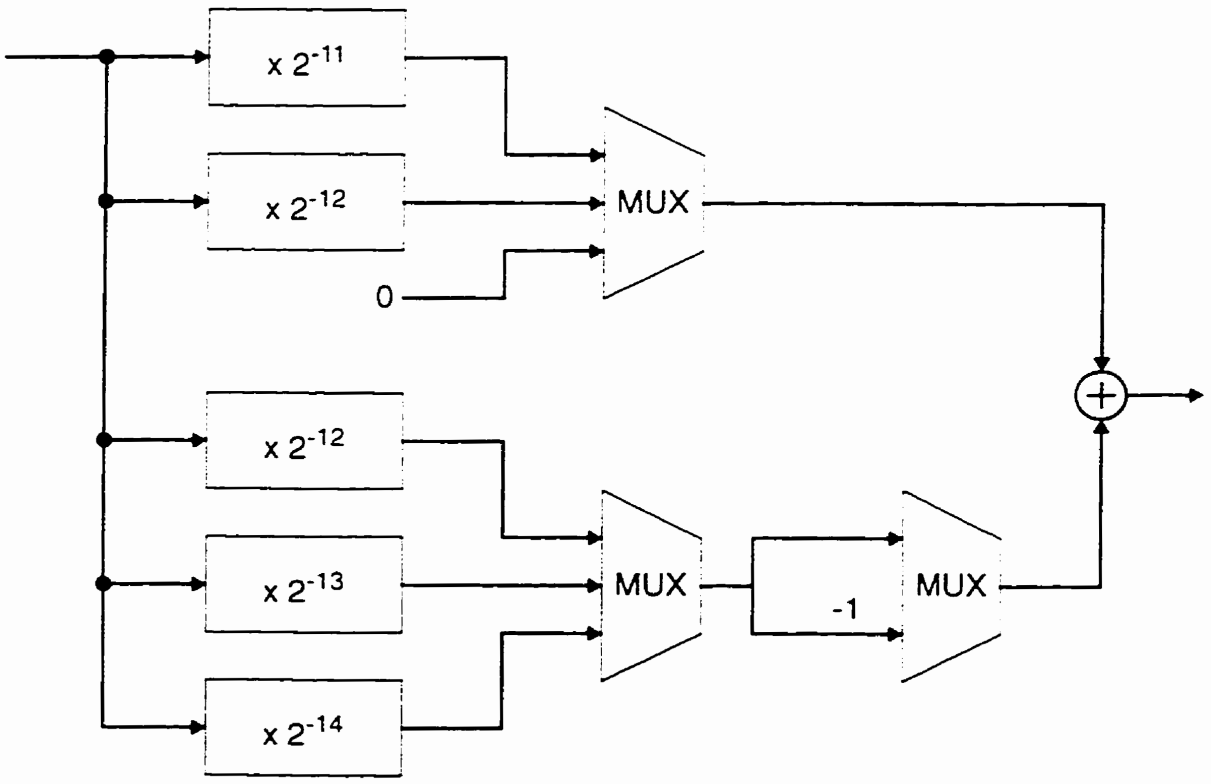


Fig. 3.5: Implementation of the programmable CSD-coefficient k_1.

the appropriate loop gain k is computed. Then the closest available value of k_1 is selected and the multiplexers in the schematic of Fig. 3.5 are set so as to implement it. Finally, k_0 is computed as the difference between k and k_1.

As an example, Table 3.8 gives the revised stability results for two values of k. Also listed are the values of k_0 and k_1. These results, when compared to those of Table 3.4, show that the use of the CSD-coefficient k_1 indeed solves the stability problems previously encountered.

Table 3.8: Revised Oscillator Stability Results for N=4, OSR=128

k	10^{-4}	$5x10^{-4}$
k_1	$1.22x10^{-4} = 2^{-13}$	$5.49x10^{-4} = 2^{-11} + 2^{-14}$
k_0	$-2.2x10^{-5}$	$-4.9x10^{-5}$
f	$1.6x10^{-3}$	$3.6x10^{-3}$
f/f_u	0.407	0.911
A_t = 0.025	A_f = 0.0252 SNR = 108.2 dB	A_f = 0.0248 SNR = 108.1 dB
A_t = 0.1	A_f = 0.100 SNR = 119.9 dB	A_f = 0.101 SNR = 118.1 dB
A_t = 0.25	A_f = 0.251 SNR = 126.4 dB	A_f = 0.253 SNR = 127.1 dB
A_t = 0.5	A_f = 0.502 SNR = 135.0 dB	A_f = 0.505 SNR = 132.4 dB
A_t = 0.75	unstable at t=4.50e+01	unstable at t=3.00e+02

This oscillator design is thus very stable, and offers a wide tone dynamic range. The latter fact is illustrated by the results of simulations for mid-band tones of amplitudes varying from 10^{-7} up to past the onset of instability, with results shown in Fig. 3.6. These simulations were again conducted over 10^8 samples. The peak SNR is approximately 130 dB.

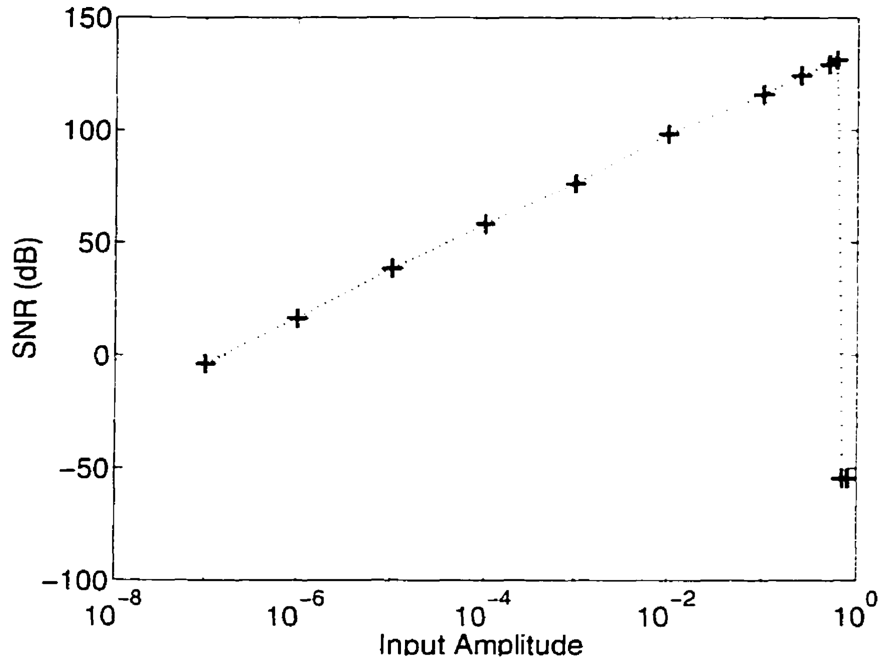


Fig. 3.6: SNR vs programmed tone amplitude for the N=4, OSR=128

3.4.3 FPGA Prototype

The stable 4th-order design presented in the previous section was to be implemented on a Xilinx FPGA, using VHDL-code generated by DSMOD, the CAD tool presented in Chapter 4. The modulator uses a 24-bit numerical format with 2 integer bits, while the oscillator uses a 38-bit register-length. Shorter registers could be used in the oscillator by scaling its internal variables, but at the cost of reduced frequency resolution. Unfortunately, once synthesized (using the Synopsys FPGA compiler) the prototype required more hardware than is contained in a single Xilinx 4010 FPGA, the prototyping platform available for this research work. However, previous experience has demonstrated that for smaller circuits the FPGA prototype behaves according to a simulation of the VHDL code as performed by the Synopsys VHDL simulator.

The inband spectrum produced by the VHDL-level simulation of this prototype for a total loop gain of 5×10^{-4} and a tone amplitude of 0.5 is displayed in Fig. 3.7. The SNR is equal to 130 dB as predicted by the DSMOD simulations.

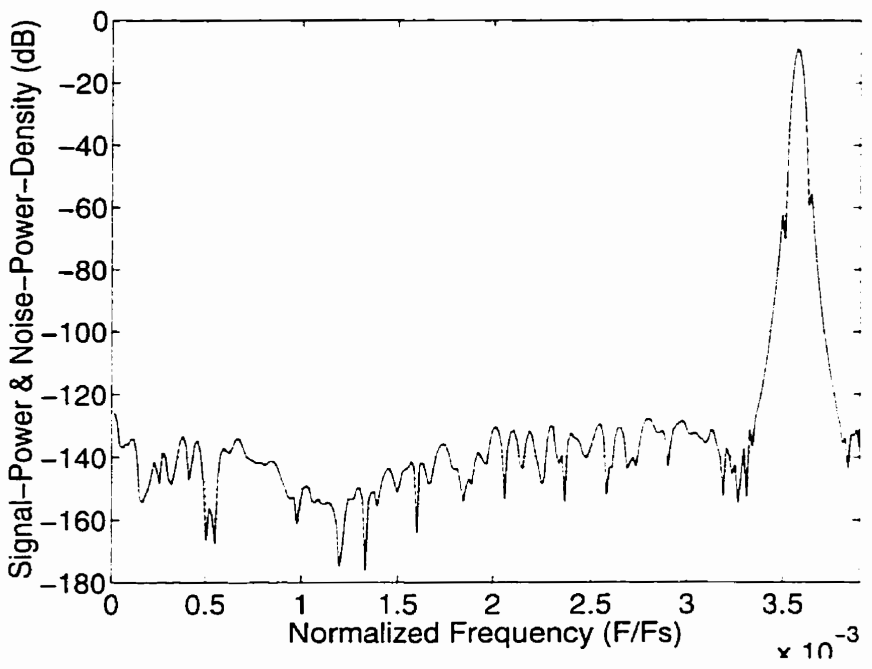


Fig. 3.7: Inband power spectrum generated by the 4th-order oscillator prototype with OSR=128.

3.5 Additional Improvements

In the context of the generation of stimuli for frequency-testing analog circuits, three additional improvements are presented. The first is the generation of tones over an arbitrary frequency band. The second is the simultaneous generation of multiple tones, and the third is the matching of the oscillator's noise spectrum to the response of the circuit under test.

3.5.1 Arbitrary Passband

So far it has been assumed that a sinusoid was to be generated at a frequency close to DC, i.e. that we were dealing with lowpass oscillators built using lowpass modulators. In many applications it may be desirable to generate tones in an arbitrary frequency band

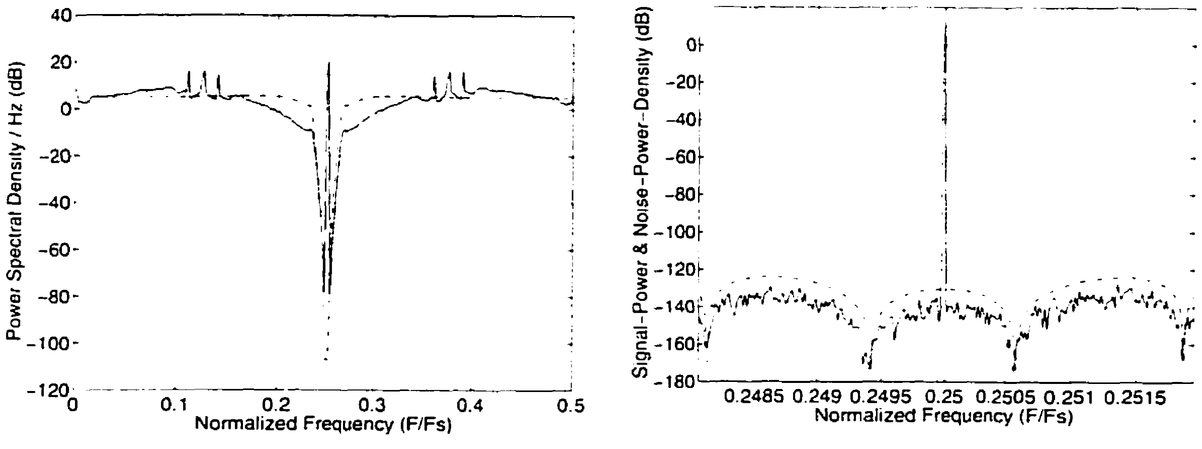


Fig. 3.8: Power spectrum of the signal generated by an 8th-order bandpass oscillator: (a) Nyquist interval; (b) signal band.

in the Nyquist interval. In that case the modulator must be designed for the same signal band; in other words a bandpass or highpass modulator must be used.

The topology for a bandpass oversampled oscillator presented in [12] and [15] can be used with a high-order bandpass delta-sigma modulator to create a high-quality bandpass signal generator. Alternatively, the topology of Fig. 3.1 can be used; k_1 must be chosen so as to keep the value of k_0 small when frequencies in the signal band are generated. The delta-sigma modulator used in the circuit must be a bandpass modulator whose signal band corresponds to the oscillator's range of tone frequencies. A bandpass modulator can be obtained from a lowpass modulator by replacing each integrator by an appropriate biquad, based on a frequency transformation equation for discrete-time systems, as explained in detail in Section 2.5.3.

Fig. 3.8 shows the spectrum generated by an 8th-order bandpass oscillator with OSR=128. The signal in this example is centered at a quarter of the sample rate, but any other center frequency is possible. The modulator is realized with quantized coefficients for hardware-efficiency. Simulations over a billion samples indicate that this design is stable. The spectral plots of the output show an SNR equal to 130 dB for a signal amplitude of 0.5. Notice the zeros of the noise spectrum in the signal band; four of the notches correspond to the zeros of the modulator NTF, while the fifth one at midband is created by the resonator loop. As a comparison, a bandpass modulator of order 4 such as the one presented in [12] provides an SNR of 85 dB for a signal amplitude of 0.5, and an OSR of 128. This examplifies the fact that increasing the order of the modulator increases the SNR of the generated tone for bandpass oscillators also. Note also that one advantage of the bandpass oscillator over the lowpass version is that no harmonics of the generated signal are present in the signal band.

Highpass oscillators are also possible. The CSD loop gain coefficient k_1 is set to a value near 4, and a highpass modulator as described in Section 2.5.3 is used.

3.5.2 Multitone Oscillators

Multitone signal generators are essential for frequency response and intermodulation tests, such as the ones presented in [3]. The principle of time-division multiplexing is used to modify a single-tone oscillator to a multitone one, as explained in [6] and [8]. To obtain an M-tone circuit, each register in the original circuit must be replaced by M registers in series, both in the resonator loop and in the modulator, as shown in Fig. 3.9. Note also that the resonator coefficients k_0 and k_1 are cycled through M distinct values, corresponding to the frequencies of the M tones. This is achieved by using a simple multiplexing scheme and multiples of the main clock signal.

Although previous studies were concerned with multitone generators using 2nd-order lowpass [6] and 4th-order bandpass modulators [12] only, the principle applies equally to higher order circuits, as first reported in [16]. A higher-order generator is capable of generating more tones at a given SNR, or alternatively it can spread the same number of tones over a wider signal band, still with the same SNR. Thus, even when extremely low noise levels are not required, higher-order oscillators can still be used to speed up the testing process by exciting an analog circuit at a greater number of frequencies simultaneously, or to test circuits over a wider bandwidth.

Fig. 3.10 shows the simulated output of a four-tone generator based on a 4th order lowpass delta-sigma modulator. The amplitude of each tone was set to 0.25 or -15 dB. Since the four tones are actually time-domain multiplexed, their effective power is divided by four, resulting in -21 dB tones as seen on the plot (the tones are actually slightly lower on the plot due to the windowing of the data before the FFT is computed).

An attractive feature of this generation scheme is that the parameters of each sinusoid (amplitude, frequency and phase) can be set independently of the other sinusoid. This is very useful to control the crest factor of the complex sinusoid used to excite the analog circuit under test, so as to avoid saturation and other non-linear effects. Note also that bandpass and highpass multitone generators are also possible.

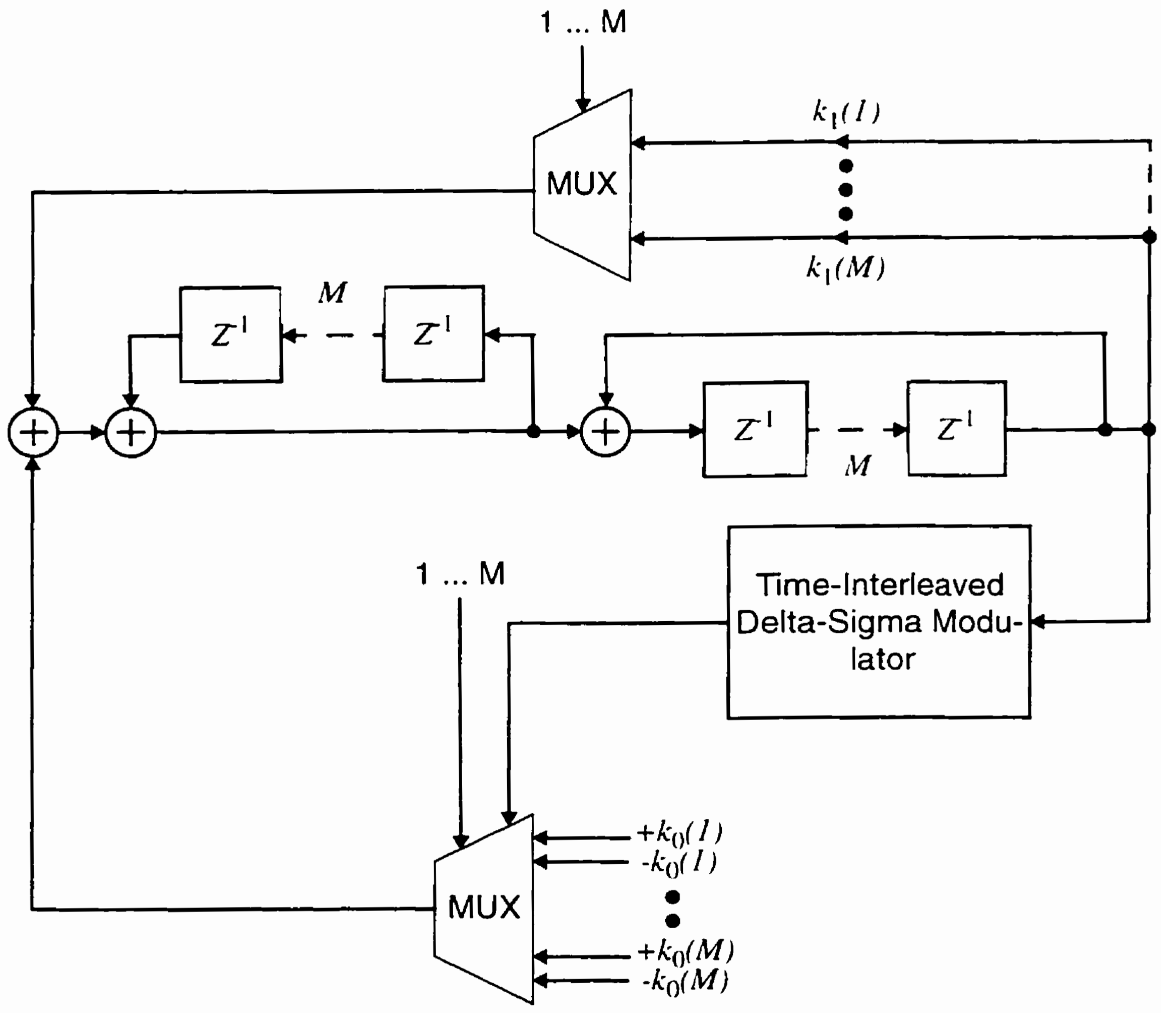


Fig. 3.9: M-tone signal generator.

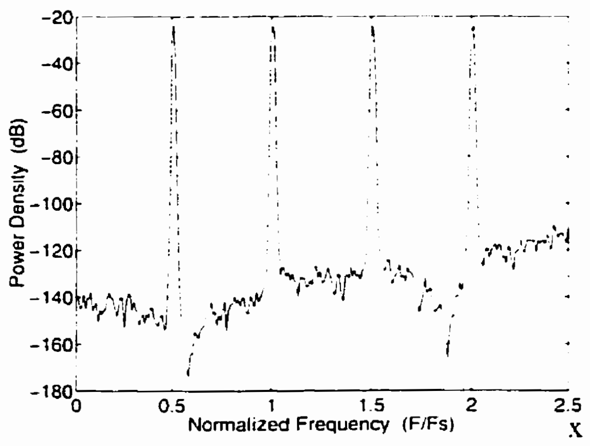


Fig. 3.10: Simulated inband spectrum of a four-tone, 4th-order lowpass oscillator.

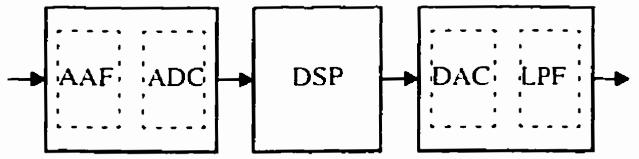


Fig. 3.11: Overall structure of a typical mixed-signal IC

3.5.3 Optimized Noise Spectrum

The additional degrees of freedom provided by a high-order design can be used to shape the out-of-band noise in the digital domain so as to accommodate the limitations of the analog circuit under test. (This concept has been proposed for the design of delta-sigma modulators in general in [46]).

To illustrate this concept we consider the situation in which the analog-to-digital converter contained in a simple lowpass codec (shown in Fig. 3.11) must undergo a sinusoidal-input based test. In many instances the anti-aliasing filter (AAF) preceding the ADC-proper will be much less sensitive to high levels of high-frequency input-noise than the ADC, since it is designed to attenuate such inputs. High noise levels at some frequencies outside the signal band could trigger non-linear effects and corresponding responses of the ADC in the signal band, which would decrease the test accuracy.

Our strategy here is to match the noise spectrum of the signal source to the transfer function of the AAF, so that the ADC sees a uniform noise power spectrum in addition to the sinewave input. In effect, this kind of noise-shaping maximizes the dynamic range of the test stimulus seen by the ADC by using the AAF as a lowpass filter.

3.6 Conclusion

High-order delta-sigma modulation-based analog signal generation has been shown to be feasible, both with respect to issues of stability and of hardware cost. The increased accuracy of high-order designs, which has been demonstrated by simulation results and experimental data, can be used to achieve various improvements in the quality and speed of on-chip testing of analog circuits. In particular, for a given signal bandwidth and digital clocking rate, high-order designs present two major improvements over lower-order designs previously presented: purer tones can be produced and more simultaneous tones of a given SNR can be produced. Also, the characteristics of the analog device being tested can be better matched so as to reduce unwanted non-linear responses to the out-of band noise associated with delta-sigma-modulation-based signal generation, which would degrade the test. All these improvements were made possible by limiting the gain k_0 in the delta-sigma modulator loop; this in turn was accomplished by including an additional, hardware-efficient scaling loop with gain k_l, which bypasses the modulator.

Chapter 4

A Computer-Aided-Design Tool: DSMOD

4.1 Motivation and Requirements

The delta-sigma modulators and oscillators presented in Chapter 2 and Chapter 3 are difficult to design by hand, because many of the design steps are computationally intensive. Some of these require solving non-linear equations (NTF design, structure mapping) while some are exhaustive searches through a solution space (coefficient quantization, oscillator design, optimal NTF design). Also, simulation is essential to determine stability and frequency-domain measures such as SNR and harmonic distortion. Finally, fast prototyping of these circuits is desirable in order to assess the effects of finite wordlength and their behavior over very long periods of time, before they are integrated onto silicon, which represents a much larger investment in time and resources than prototyping. Quick prototyping is possible with Field-Programmable-Gate-Arrays (FPGAs), which can be synthesized with a Hardware-Description-Language (HDL) compiler. Hand-coding the HDL description of a circuit requires technical knowledge of the language and may take some time to debug, and this can advantageously be automated too.

These facts motivated the creation of a computer-aided design (CAD) tool to implement design, simulation, and prototyping of digital delta-sigma modulators and oscillators of arbitrary order and signal band. This tool bears the name DSMOD for "Delta-Sigma Modulator and Oscillator Designer".

Section 4.2 of this chapter covers the existing CAD tools which solve problems similar to the ones at hand, and discusses their applicability to the design of delta-sigma oscillators. Based on that brief discussion, Section 4.3 explains the choices that were made regarding the actual implementation of DSMOD. Section 4.4 presents each of the capabilities of the tool in detail, including a description of the user interface and of the algorithms. Possible improvements to the tool are discussed in Section 4.5.

4.2 Existing CAD Tools

Numerous general-purpose filter and circuit design tools exist but they are not reviewed here because they cannot provide an integrated environment used to design, simulate and implement digital delta-sigma modulators and oscillators. More specialized tools which deal with specific aspects of these three tasks are regularly reported in the literature. FiltorX [46], for instance, can design continuous time filters of any order, according to a variety of optimization criteria and against arbitrary sets of constraints on the magnitude and phase response. This tool could be used to design a NTF. The shortcoming is that the rest of the design and simulation work must still be performed by another tool. However when a very special NTF is needed (such as for optimizing the noise spectrum of a signal source in a mixed-signal testing application, as proposed in Section 3.5.3), a tool such as FiltorX should be used, and the resulting NTF can then be passed on to the delta-sigma oscillator CAD tool.

Another design step for which a number of tools exist is coefficient quantization, or digital filter design with CSD coefficients. Many such tools optimize the design of finite-impulse response (FIR) filters with CSD coefficients [40][41][42][43]. These filters have a large number of zeros (sometimes more than 100) and the tools generally seek to minimize the maximum ripple in the passband and stopband. The linear filters embedded in delta-sigma oscillators are generally infinite-impulse response (IIR) filters, and therefore the aforementioned these tools cannot be used. In addition it may be difficult to integrate these specialized tools with our delta-sigma oscillator design tool.

DSMOD, which is the focus of the remainder of this chapter, was first presented in [18] as an integrated design and prototyping environment for delta-sigma oscillators.

4.3 Implementation Choices

4.3.1 Programming Platform

The preceding section points to the fact that even though a specialized CAD tool is powerful to solve part of the design problem, it may be difficult to link it to the rest of the design flow. For this reason the tool presented here was programmed for MATLAB [44], due to the flexibility and widespread use of this computation and visualization software package. In other words, MATLAB constitutes the common basis through which a number of tools can communicate at a high-level of abstraction by sharing variables and functions. In addition, MATLAB handles all the computational aspects of the design problem and can be supplemented with a variety of toolboxes containing high-level functions such as transfer function design and non-linear equation solving, as well as graphical user interface (GUI) capabilities, thereby greatly reducing the difficulty of the task of programming the CAD tool. Finally, MATLAB is an interactive software as well as a programming language, and thus debugging the tool is made easy by direct access to all the variables and functions. The only drawbacks are that MATLAB has little provisions for object-oriented programming, has no data structures except arrays, and its language is interpreted rather than compiled.

Simulations require simpler mathematical operations since they implement simple difference equations. Thus they are better implemented in a compiled language. C was chosen for this purpose, given its widespread use across all computer platforms, and even as a functional modelling language in some instances, for example for Digital Signal Processors. The simulators are interfaced with the rest of the tool via data files using a predetermined format.

4.3.2 Graphical User Interface

The advantages of graphical user interfaces (GUIs) need not be proven here, as virtually all of today's software incorporate one. The CAD tool should make the design flow explicit, and let the user see only the relevant design parameters. The result is that no time is wasted wondering what is the next design step and what parameters are important. Rather, any extra time can be used to evaluate the impact of a wider variety of design choices.

A CAD tool can also be useful as a teaching tool, since it exposes the user to all aspects of the design. Actually, since the design of delta-sigma modulators and oscillators cannot be done by hand, due to the complexity of the task, learning to design them amounts to learning how a computer can design them. This learning process would be impeded if a specific language had to be learned in order to do the design and visualize the results. With a graphical interface, the user focuses solely on the design steps, parameters and results (transfer function plots, coefficient values, circuit topologies, simulation outputs) without getting involved in producing all these results. DSMOD has been used at McGill University in the graduate VLSI design project course.

Examples of the GUI will be shown in the next section, as they relate to each individual module of the CAD tool.

4.4 Design Flow and Algorithms

Let us reiterate the problem to be solved by the oscillator designer using DSMOD. Given specifications on the signal bandwidth and on the SNR of the generated tone over the signal band, a detailed signal flow graph (SFG) of a delta-sigma oscillator must be obtained and validated through simulation and prototyping. This signal flow graph may look like the one shown in Fig. 4.1. It consists of two main digital blocks: a Lossless-Discrete-Integrator (LDI) resonator and a 1-bit digital delta-sigma modulator (here a lowpass, LDI-ladder based modulator is depicted, although modulators of other topologies and passband types could be used; refer to Section 2.5.2 and Section 2.5.3). The modulator is of arbitrary order N. All the scaling coefficients except k_0 (A_1 to A_N, B_1 to B_N and k_1 to

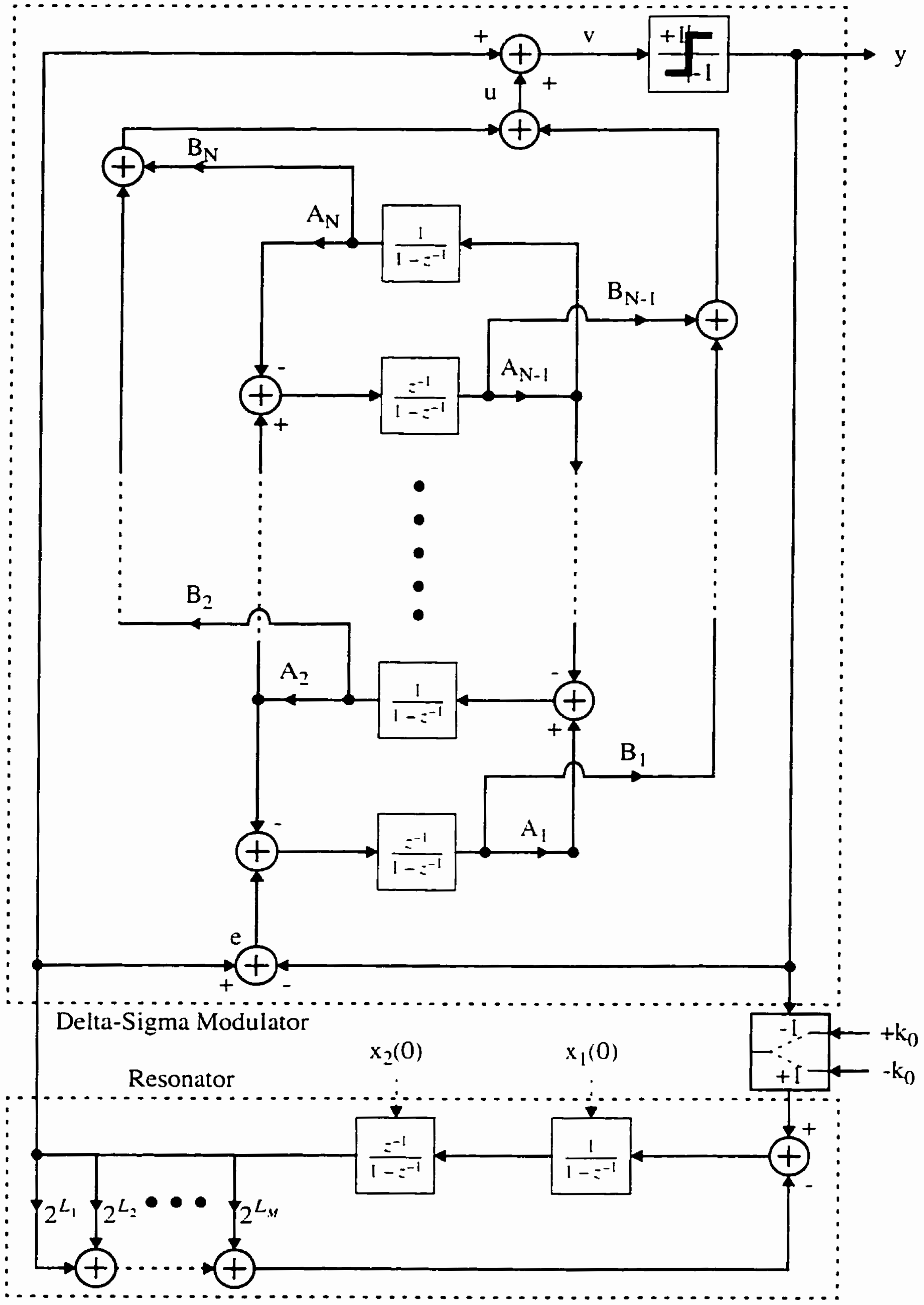


Fig. 4.1: Generalized signal-flow graph of the delta-sigma oscillator.

k_M) are CSDs (sums/differences of powers of two) and so are realized by shift units and adders instead of multipliers. As demonstrated in Chapter 3, when all the coefficients and the initial values of the resonator integrators ($x_1(0)$ and $x_2(0)$) are properly chosen, the circuit resonates and creates a one-bit output whose power density spectrum is composed of a single tone and noise concentrated at frequencies outside the signal band. The design objectives are to determine the modulator topology, the order of the modulator (i.e. the number of integrators in the modulator), a set of cheaply realizable modulator coefficients and a set of power-of-two resonator coefficients (2^{L_1} to 2^{L_M}) which implement the programmable coefficient k_1 so as to ensure that the oscillator is stable and produces tones of the required SNR over the given signal band.

The next section describes the entire design flow from a global perspective. The following sections explain each design step in detail. In each case the graphical interface and design parameters are presented, the relevant algorithms are explained, and the outputs (text, plots and files) created by the tool are described.

4.4.1 Overview of the Design Flow

The delta-sigma modulator used in the oscillator sets the resolution and bandwidth of the oscillator's output signal. Thus, as shown in Fig. 4.2, the design flow starts with the creation of a delta-sigma modulator (represented by the grey box), more specifically by specifying the signal band and designing a modulator NTF which results in an appropriately low noise floor in the signal band. An arbitrary NTF can be designed, or alternatively DSMOD can search for an NTF which maximizes the SNR of the modulator output (note that it is hard to predict the input level at which the maximum SNR is reached, thus the need to perform a search). In either case the result is a pair of polynomials or equivalently a set of poles and zeros representing the desired NTF.

At this point the user can elect to either design an oscillator or to continue the modulator design procedure. The latter option requires that the user select a modulator topology, for which DSMOD computes the coefficients which realize the desired NTF (shown as "Structure Mapping" in Fig. 4.2).

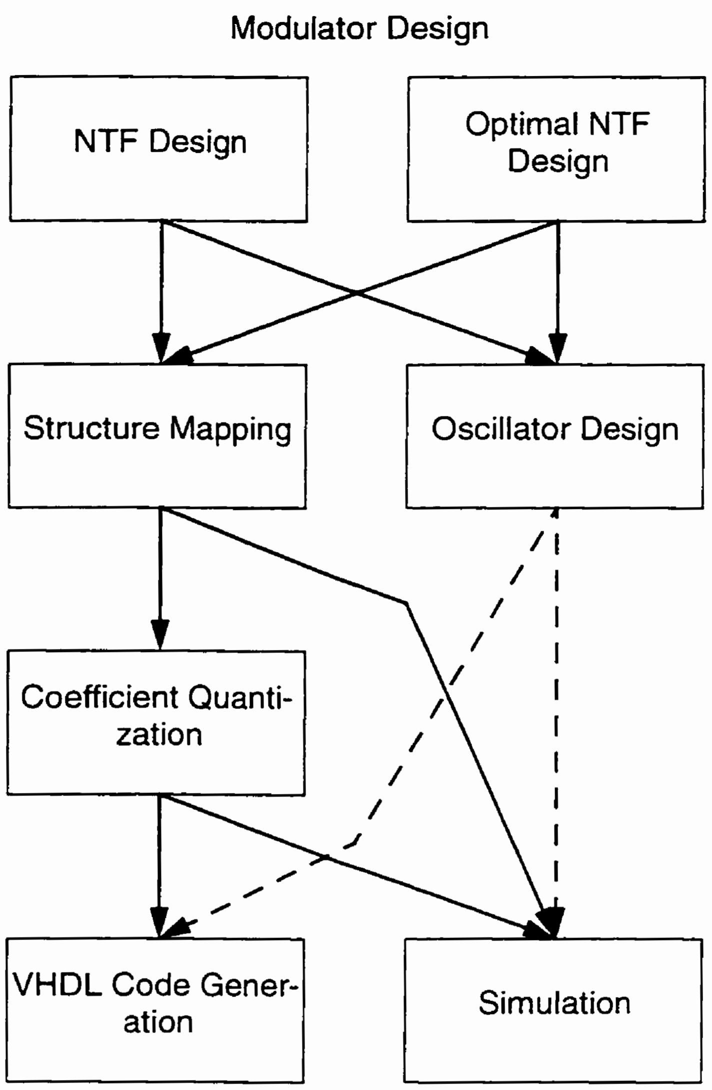
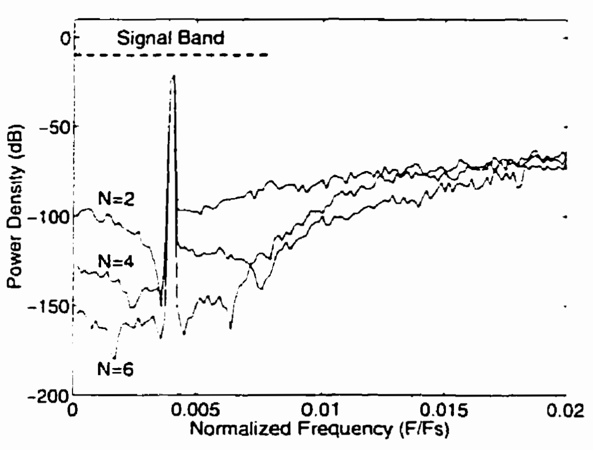


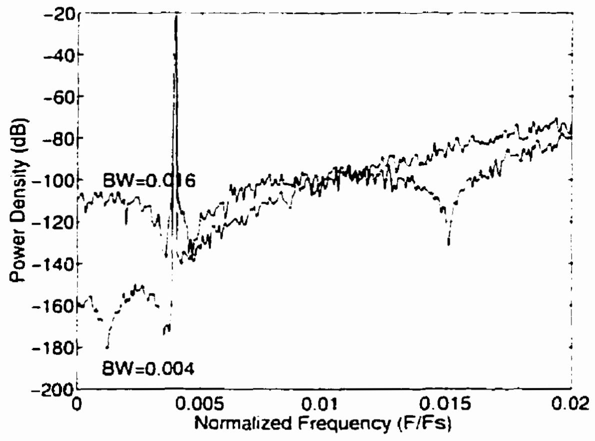
Fig. 4.2: Overview of the DSMOD's design flow.

After the modulator coefficients have been computed, the tool can perform a simulation of the modulator or of the oscillator (if one has been designed, as implied by the dashed arrows), which is needed to establish stability, SNR, and distortion level. Another available option is to quantize the coefficients to CSDs, which also leads to the possibility of simulating the resulting design.

Designing the oscillator amounts to computing a minimal set of power-of-two values of 2^{L_1} to 2^{L_M} which ensure that tones can be generated at frequencies throughout the signal band while k_0 is kept below a threshold specified by the user. The modulator need



(a)



(b)

Fig. 4.3: (a) Power spectra of the simulated output of oscillators using delta-sigma modulators of order 2, 4 and 6; (b) using a 6th-order modulator, for two different bandwidths.

not be designed for this step; only the signal band must be known by the tool. However, simulating the oscillator (and thus proving its stability) requires that a modulator have been designed up to "Structure Mapping".

Lastly, a prototype can be generated only if the modulator coefficients have been quantized. Again, either a modulator or an oscillator prototype can be generated.

4.4.2 NTF Design

Fig. 4.3(a) shows examples of power spectra for oscillator designs using modulators of various orders N. It illustrates one of the basic design trade-offs involved here: modulator order vs. dynamic range. It can be seen that higher-order designs result in a lower noise-floor in the passband, while they require more hardware. Using DSMOD, one can quickly find the modulator order needed to meet the design specifications. Another trade-off is that of bandwidth vs. dynamic range. If the range of signal frequencies is extended relative to the digital clock rate F_s and the order of the modulator kept constant, then the noise floor is raised, as illustrated by Fig. 4.3(b). However, if the physical bandwidth is fixed, this means that a slower digital clock rate can be used. The NTF design module lets the user explore and optimize both these trade-offs.

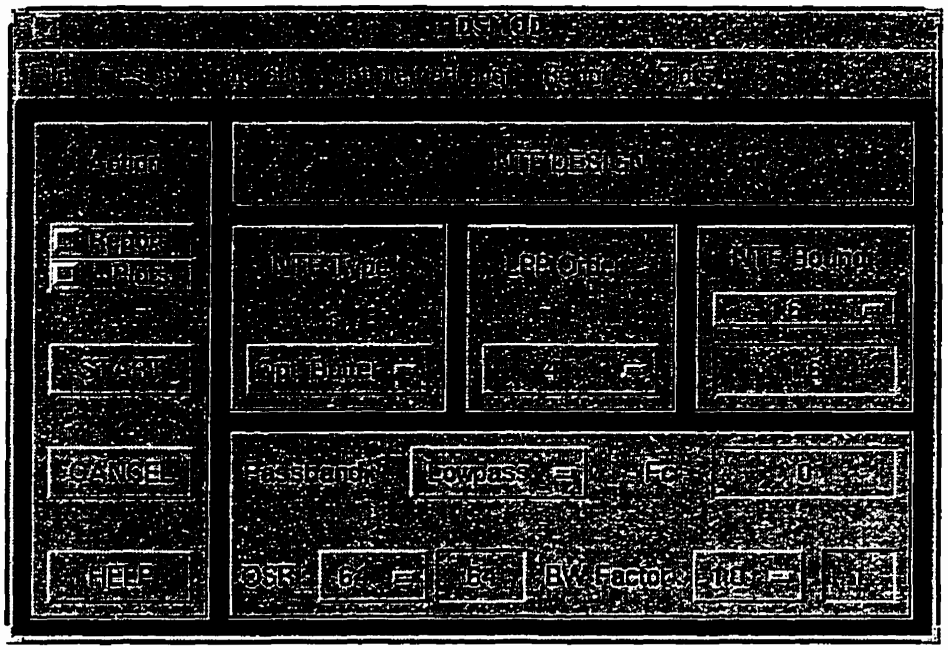


Fig. 4.4: **Dialog box for the NTF design module.**

The user-interface for the NTF design is shown in Fig. 4.4. The design parameters can be subdivided in two sets. The passband type, the oversampling ratio (OSR) and the center frequency F_c define the signal band. The other parameters determine the shape of the NTF. The NTF can be elliptic, butterworth with optimally-located zeros, or supplied by the user; this is controlled by the 'NTF Type'. The order of the NTF is given by 'LPP Order' if it is lowpass or highpass, or twice 'LPP Order' if it is bandpass. Finally, the NTF Bound allows for controlling the stability of the modulator. It can be set to any value between 1.0 and 2.0; values closer to 2.0 yield modulators with a higher inband noise attenuation but a smaller range of stable input amplitudes. Finally, the bandwidth factor ("BW Factor" in the dialog box) allows for designing a NTF attenuating the quantization noise over a smaller or larger band than the actual signal bandwidth; this may be desired to adjust the design against subsequent alterations of the NTF due to coefficient quantization effects.

The "Opt. Butter." NTF type stands for "Butterworth with optimally located zeros". This design option produces the best NTFs, in our experience. The design algorithm designs a lowpass prototype NTF as follows. Let us first recall that, as stated in Sec-

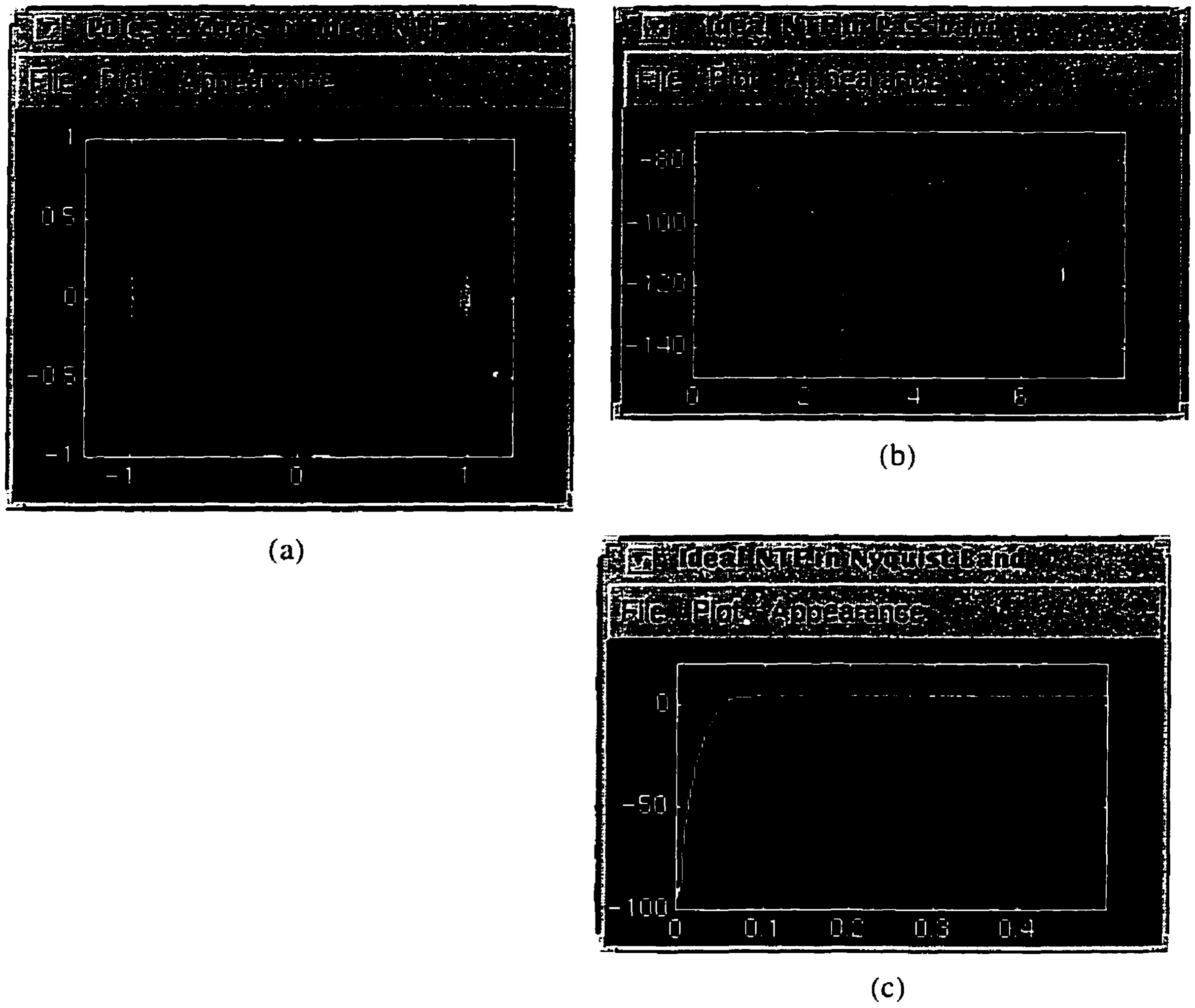


Fig. 4.5: Plots created by the NTF design module: (a) Pole-Zero plot, (b) Inband plot of the NTF magnitude, (c) Nyquist-band plot of the NTF

tion 2.5.1, the NTF numerator and denominator must have equal leading coefficients. DSMOD designs a succession of highpass butterworth transfer functions with that property: the only degree of freedom is the natural frequency of the transfer function. As the natural frequency increases above 0, the maximum value of the NTF magnitude increase above 1.0. The procedure is stopped when the NTF bound has reached the desired value. The poles of the resulting transfer function are kept, and the zeros at DC are replaced by zeros on the unit circle, optimally located in the signal band so as to maximize the average inband attenuation. The optimal zero locations are taken from [25]. If the desired NTF is bandpass or highpass, the poles and zeros are then mapped as explained in Section 2.5.3.

Fig. 4.5 shows the graphical output of the NTF design operation. Three plots are created; the first displays the poles (x's) and zeros (o's) of the designed NTF, the other two show the NTF magnitude response over the signal band and over the Nyquist interval, that is from DC to half the sampling rate.

4.4.3 Structure Mapping

DSMOD offers a choice of three modulator topologies, namely the integrator cascade, the resonator cascade, and the LDI-ladder. These are shown in Fig. 2.11, Fig. 2.15 and Fig. 2.15, respectively, in Section 2.5.2. Once the user has selected one of these three topologies (via a trivial dialog box, not shown here for the sake of conciseness), the tool computes the A and B coefficients for which the modulator will have the desired NTF. This is done by simultaneously solving the equations expressing each term of the NTF numerator and denominator in terms of the modulator coefficients. It turns out that for the three topologies the numerator of the NTF depends only on the A coefficients, and thus the procedure of solving the equations is broken up in two distinct steps, first for the A coefficients, then the B coefficients.

For instance, the equations relating the NTF of a 4th-order lowpass LDI-ladder modulator to its coefficients are given below:

$$NTF(z) = \frac{z^{-4} + a_1 z^{-3} + a_2 z^{-2} + a_1 z^{-1} + 1}{b_4 z^{-4} + b_3 z^{-3} + b_2 z^{-2} + b_1 z^{-1} + 1}, \tag{4.1}$$

where

$$a_1 = A_1A_2 + A_2A_3 + A_3A_4 - 4, \tag{4.2}$$

$$a_2 = (A_3A_4 - 2)A_1A_2 - 2A_2A_3 - 2A_3A_4 + 6; \tag{4.3}$$

and

$$\begin{bmatrix} 1 & A_1 & 0 & 0 \\ A_2A_3 + A_3A_4 - 3 & A_1A_3A_4 - 2 & A_1A_2 & A_1A_2A_3 \\ 3 - A_2A_3 - A_3A_4 & A_1 & -A_1A_2 & 0 \\ -1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \\ B_3 \\ B_4 \end{bmatrix} + \begin{bmatrix} a_1 \\ a_2 \\ a_1 \\ 1 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix}. \tag{4.4}$$

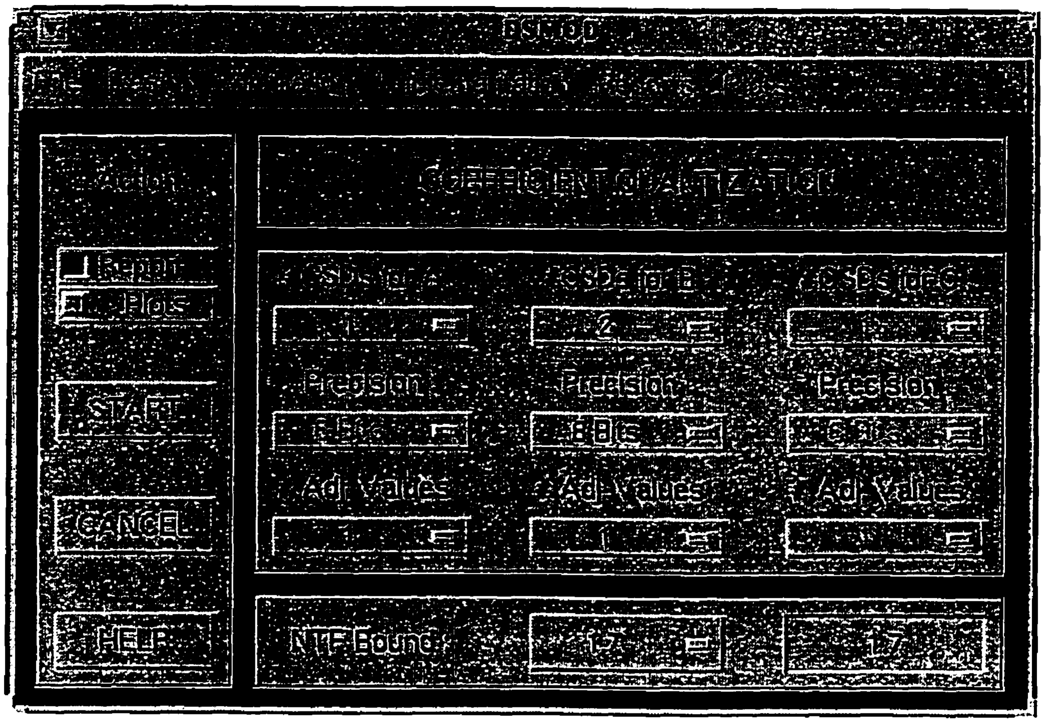


Fig. 4.6: Dialog box for the coefficient quantization module.

The tool solves the non-linear expressions given by Eqns. (4.2) and (4.3) simultaneously using MATLAB's optimization toolbox, and thus obtains the A coefficients. It then solves the linear matrix equation given by Eqn. (4.2) so as to obtain the B coefficients. Similar equations describing all three topologies, for orders 2 to 6, are programmed into DSMOD.

If the modulator is highpass, the same equations apply, and the integrators are replaced by differentiators, as explained in Section 2.5.3. If it is bandpass, the integrators are replaced by biquads described by a single coefficient C, and the equation for this biquad coefficient, Eqn. (2.16), is solved in addition to the lowpass equations.

4.4.4 Coefficient Quantization

Fig. 4.6 shows the dialog box for the coefficient quantization operation, and Fig. 4.7 represents the algorithm. Each of the coefficients can be quantized to a specified number of power-of-two terms, from 1 to 6 ("#CSDs for A/B/C"). If more than 1 CSD term is chosen, then the precision of the quantized coefficient values to be used is specified in bits. That is, if the precision is 8 bits, then the CSD terms for a single coefficients will not differ

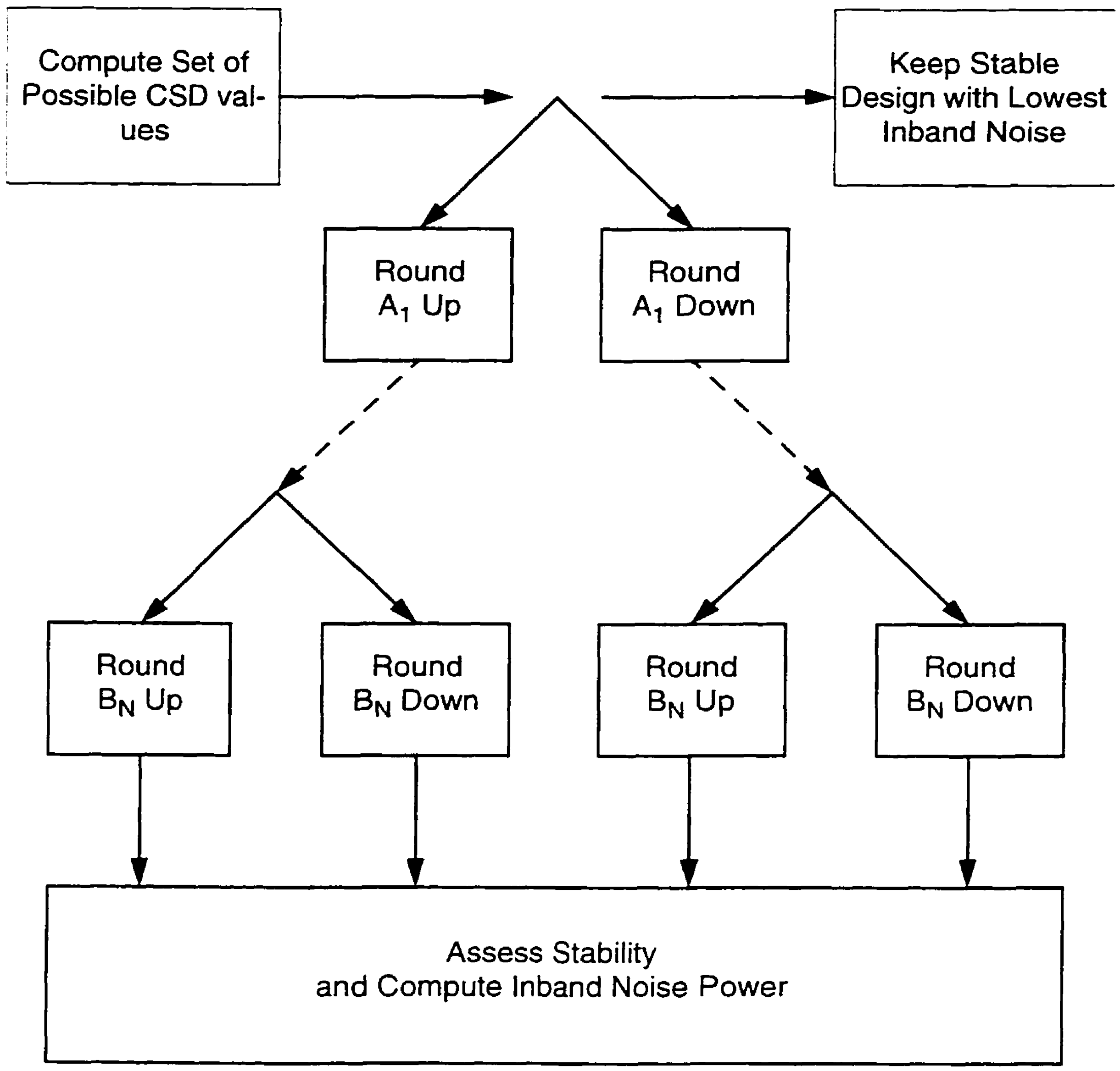
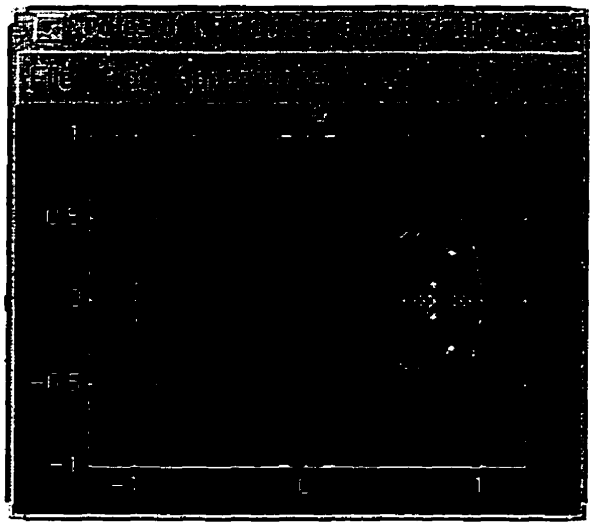
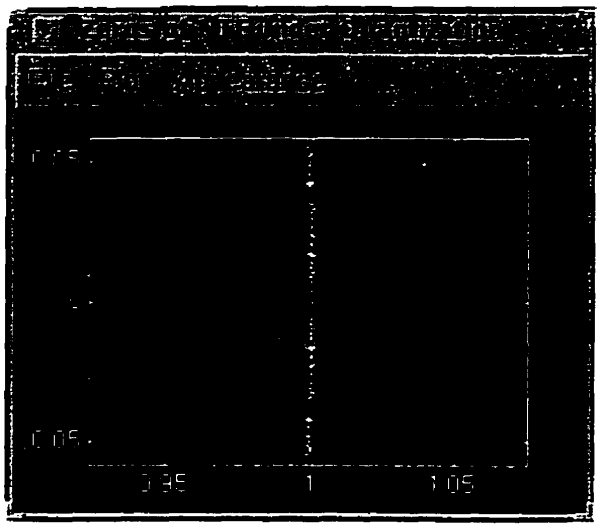

Compute Set of Possible CSD values
Keep Stable Design with Lowest Inband Noise
Round A_1 Up
Round A_1 Down
Round B_N Up
Round B_N Down
Round B_N Up
Round B_N Down
Assess Stability and Compute Inband Noise Power

Fig. 4.7: Algorithm for coefficient quantization.

by more than a factor of 2^8. These two parameters control the initial step of the algorithm, in which the allowed CSD values are computed. The parameter "# Adj. Values", on the other hand, controls the extent of the search. That is, for each coefficient, as shown in Fig. 4.7, the specified number of CSD values directly above and below the initial coefficient value are tried; if 1 adjacent CSD value is specified, the search will explore fewer quantized coefficient combinations; if more adjacent CSD values are specified, the search will be broader. Each possible set of quantized coefficients is evaluated for stability and inband

(a) (b)

Fig. 4.8: Plots created by the coefficient quantization module: (a) NTF poles under quantization, (b) NTF zeros under quantization.

noise power. The last parameter controlling the search, the NTF bound, fixes an upper bound on the maximum of the NTF magnitude resulting from the coefficient quantization process. Coefficient sets resulting in NTFs which violate this bound are rejected. When the search is complete, the coefficient set resulting in the stable modulator NTF with the lowest inband noise power is kept.

Feedback on the search is given via two plots showing the behavior of the NTF poles and zeros under the coefficient quantization process. These are shown in Fig. 4.8(a) and (b), respectively. The stars (*) represent the initial pole/zero location, while the 'x's and 'o's represent the poles and zeros of all the possible NTF's that were explored. These plots give useful feedback about the effectiveness of the coefficient quantization process. If no 'x's are found near the original NTF pole locations, for instance, then the quantization process will not yield any usable coefficient set, and the user is made aware of the necessity of increasing the number of CSD terms per coefficient, of increasing the extent of the search, or of using a different modulator structure.

4.4.5 Oscillator Design

Once the signal band has been specified, the design of a delta-sigma oscillator, apart from the modulator design, amounts to selecting the set of discrete values that k_1 must be able to take on, and how to implement these values efficiently. The principles of this design problem are explained in Section 3.4. The user specifies the maximum value that the multiplexed coefficient, k_0, can take; this bound is called k_{0max}. Recall that, according to Section 3.3, k_{0max} controls the stability of the oscillator. Any two adjacent values of k_1 cannot differ by more than $2k_{0max}$, and the goal becomes finding a small set of values of k_1, implemented using a minimum number of CSDs, for which this constraint is respected. The smaller is k_{0max}, the more stable the oscillator, but also the larger the set of values of $k_1$will be and the more costly it will be to implement.

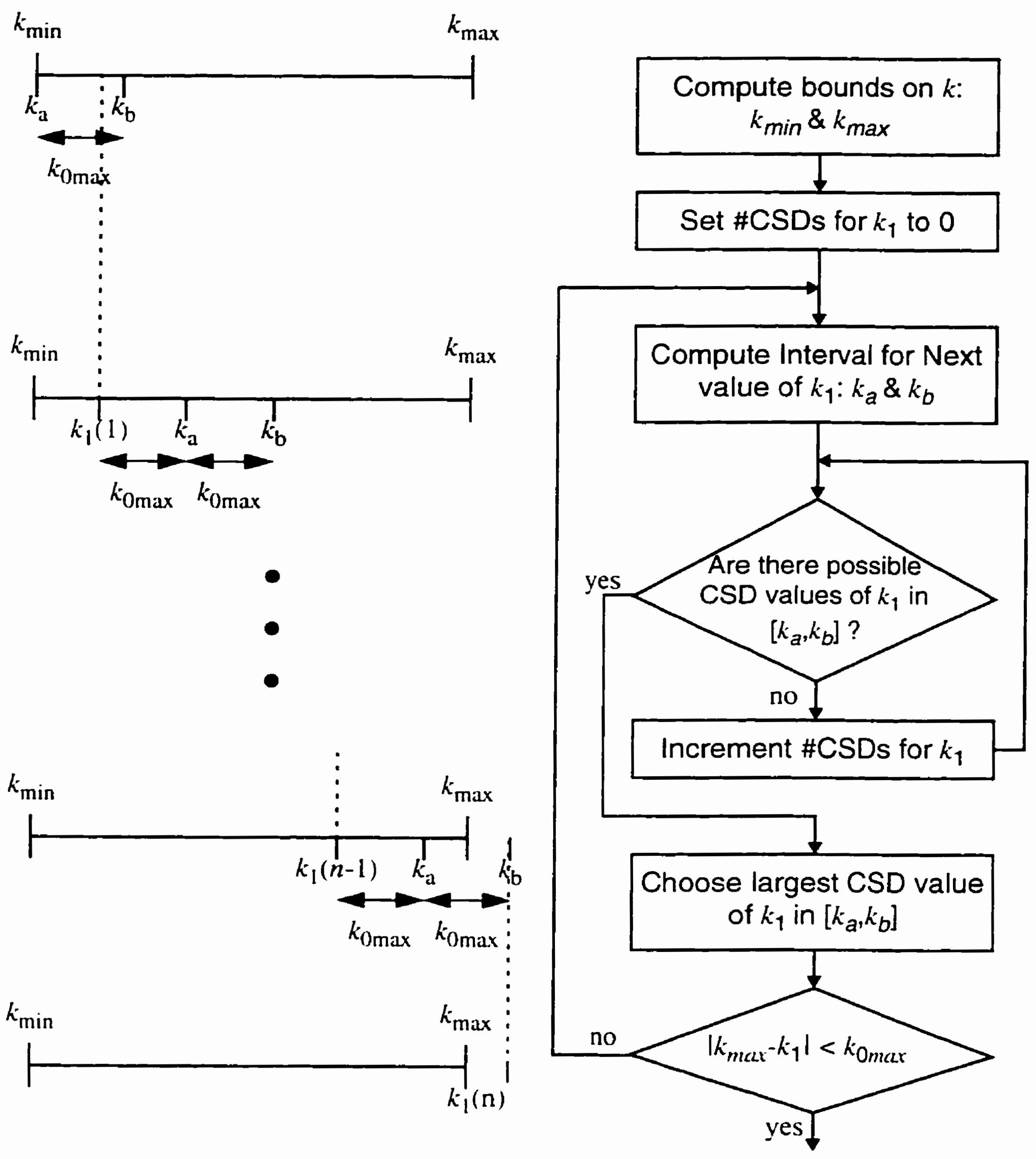
DSMOD tackles the problem using the algorithm depicted in Fig. 4.9. The range of values which the total loop coefficient k must be able to take on, $[k_{min}, k_{max}]$, is computed from the lower and upper edge frequencies defining the signal band. The first value of k_1, which we call $k_1(1)$, must fall between k_{min} and $k_{min} + k_{0max}$. If zero CSD (i.e. k_1=0) is not enough to find such a value, more CSDs are added until this value, $k_1(1)$, is found. Then the next interval in which the algorithm seeks to find a new value of k_1, i.e. $k_1(2)$, is computed; this interval is equal to:

$$[k_a, k_b] = [(k_1(1) + k_{0max}), (k_1(1) + 2k_{0max})]. \qquad (4.5)$$

The search procedure is repeated until a value of k_1 has been found which lies within k_{0max} of k_{max}, the upper bound on the loop gain k. The output of this entire operation is a table of values of k_1, expressed as sums and differences of powers-of-two.

4.4.6 Simulation

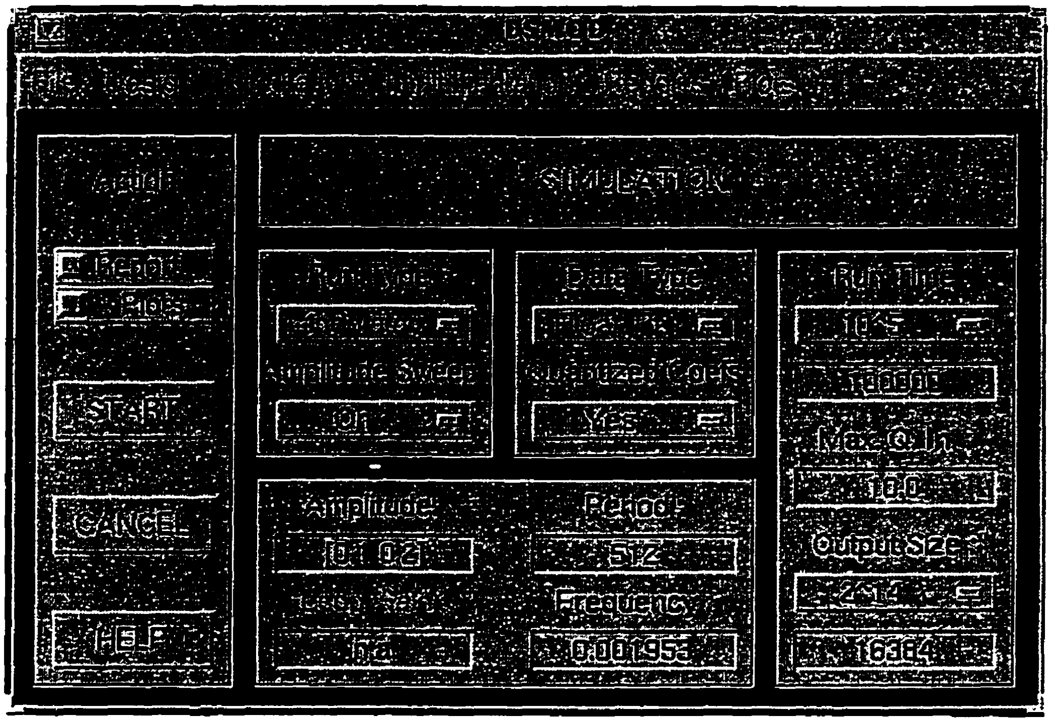
Simulations allow the user to assess whether a design is stable for a given signal level and frequency. The motivation for this capability is that delta-sigma modulators, being non-linear circuits, are only conditionally stable. When embedded in oscillator circuits, it becomes very difficult to predict their stability properties. Simulations thus serve to validate a given design in a few minutes.



k_{min}
k_{max}
k_a
k_b
k_{0max}
$k_1(1)$
$k_1(n-1)$
$k_1(n)$
Compute bounds on k: k_{min} & k_{max}
Set #CSDs for k_1 to 0
Compute Interval for Next value of k_1: k_a & k_b
Are there possible CSD values of k_1 in $[k_a,k_b]$?
yes
no
Increment #CSDs for k_1
Choose largest CSD value of k_1 in $[k_a,k_b]$
$|k_{max}-k_1| < k_{0max}$
no
yes

Fig. 4.9: Algorithm for oscillator design with a bound on k_0.

The tool includes a simulation library, capable of producing the outputs of all the modulator and oscillator topologies that it can design. If finite-register-length effects are not considered, simulating these circuits amounts to iterating through simple difference



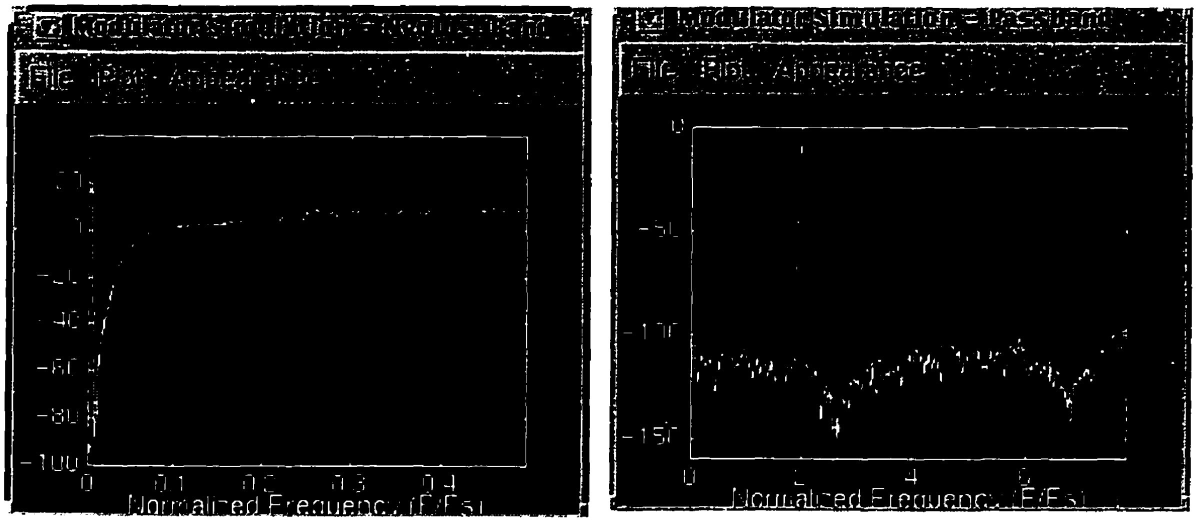
Run Type: [illegible]
Amplitude Sweep: On
Data Type: [illegible]
Quantized Coefs: Yes
Run Time: 10^5
100000
Max Q In: 10.0
Amplitude: [0.1 0.2]
Period: 512
Frequency: 0.001953
Output Size: 2^14
16384

Fig. 4.10: Dialog box for simulation.

equations. This is easily programmed in the C language; the advantage over simulating using MATLAB is a tremendous gain in speed because the simulators are compiled instead of interpreted. The interface between the MATLAB platform on which most of DSMOD runs and the simulators consists in a set a files of a predetermined format, through which the simulation parameters and results are communicated.

The simulation parameters are set by the user via the dialog box shown in Fig. 4.10. A typical simulation will produce plots of the output's power spectral density such as the ones shown in Fig. 4.11, along with a computed estimate displayed as a dotted curve.

Note also the possibility of performing a sweep of the signal amplitude. This feature can be used to quickly obtain the maximum SNR that a design can achieve. Since one cannot predict the input level at which this maximum is attained, it is read from a plot of the circuit's SNR versus the input level based on simulation results, as shown in Fig. 4.3.



(a) (b)

Fig. 4.11: Plots created from simulation output: (a) Nyquist-band power spectral density, (b) Inband power spectral density.

4.4.7 Optimal NTF Design

The functions of NTF design, structure mapping, and amplitude-sweep simulation can be combined to design an NTF which maximizes the SNR for a given modulator order and OSR. The algorithm is a basic search controlled by the parameters of the dialog box shown in Fig. 4.13. These are: the NTF bounds to design NTFs for, the signal amplitudes to simulate the designs with, the simulation time, the sample size over which the SNR is computed, the signal band parameters, and finally the modulator order. Fig. 4.14 illustrates this algorithm: NTFs are designed for a range of NTF bounds, using "Butterworth with optimal zeros" method. For each NTF, the coefficients of an arbitrary modulator structure are computed (DSMOD makes use of the resonator cascade for this purpose) and an amplitude-sweep simulation is performed. The NTF which yields the highest SNR is returned as the optimal solution for the given search parameters.

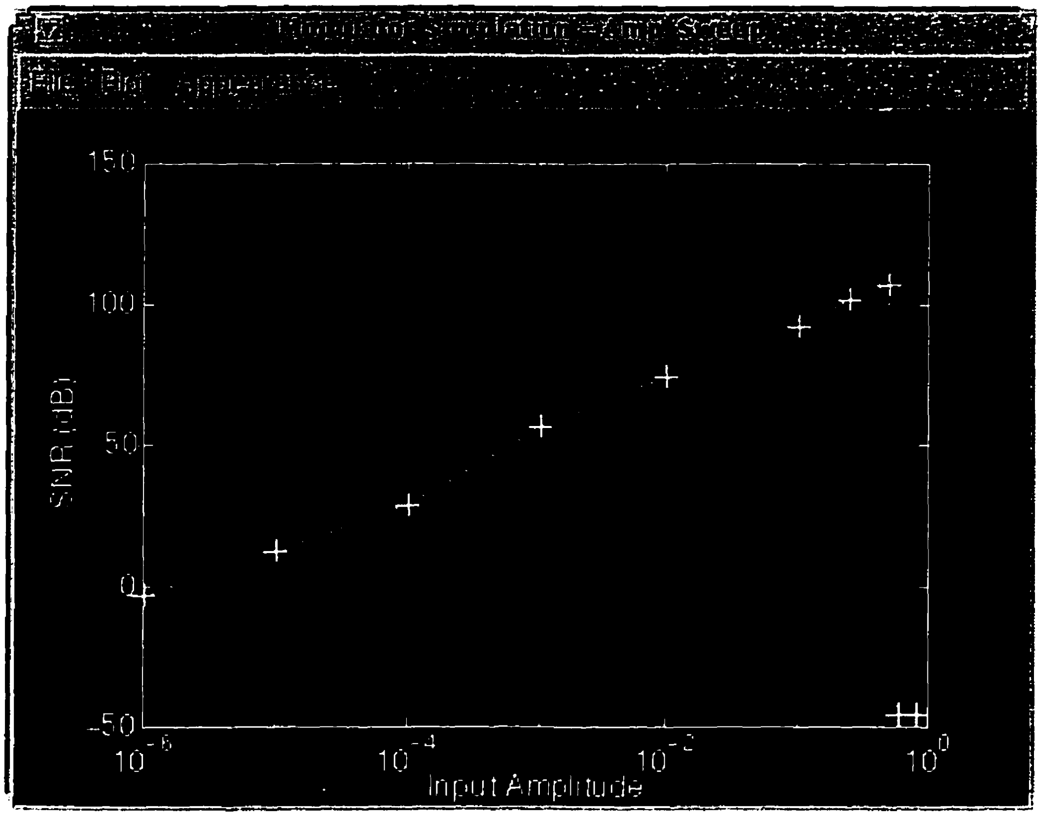


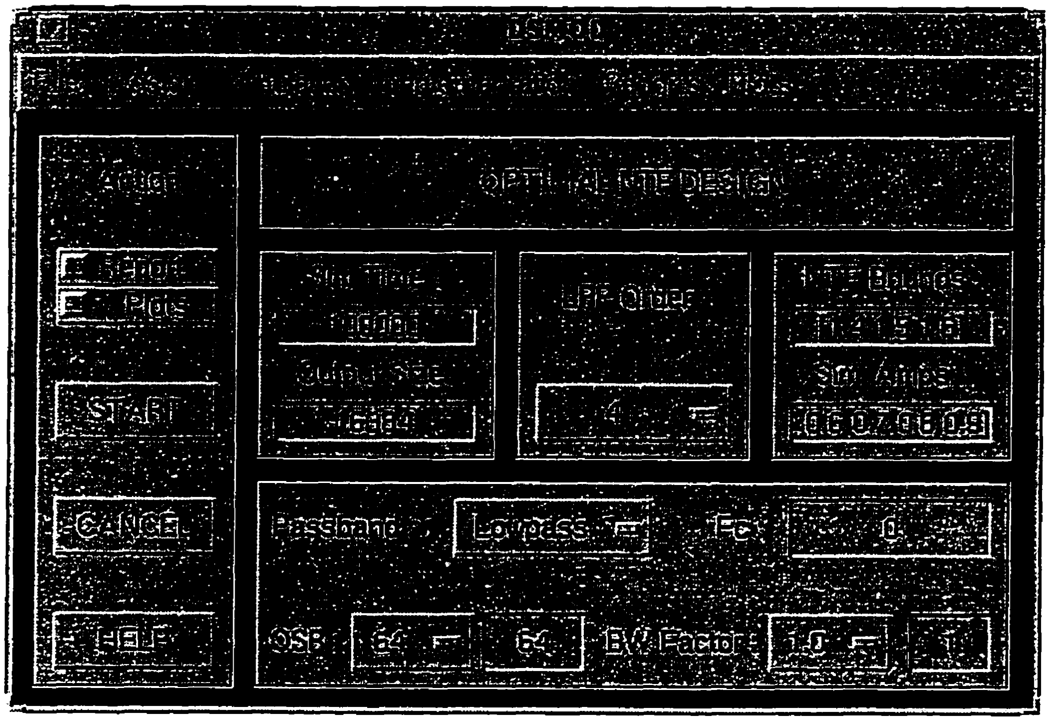
Fig. 4.12: Plot created from the output of an amplitude-sweep simulation.

4.4.8 FPGA Prototyping

Floating point simulations are very practical on a workstation but they may not reveal limitations of actual implementations (usually based on fixed-point arithmetic). In particular, some designs may become unstable over very long periods of operation, and the SNR performance may be impaired by finite-register-length effects. The tool addresses this problem by generating and testing actual prototypes of the designed oscillator.

First, the required register-lengths are entered by the user in the dialog box shown in Fig. 4.15. VHDL code is then automatically generated, along with the scripts needed to simulate it and then compile it using an external synthesis tool (Synopsys) for a Field-Programmable-Gate-Array (FPGA) technology (Xilinx 4010).

In the case of the LDI ladder structure, the integrator variables can be scaled so as to make optimal usage of the available register bits. Simulations are performed to estab-



Sim Time: 10000
Output Size: 16384
LPF Order: 4
NTF Bounds: 1.4 1.5 1.6
Sim Amps: 0.6 0.7 0.8 0.9
Passband: Lowpass
Fc: 0
OSR: 64 64
BW Factor: 1.0 1

Fig. 4.13: Dialog box for the optimal NTF design.

lish the maximum value that each register must be capable of holding. These values are then rounded up to the nearest power of two. If these values are called 2^{m_1} to 2^{m_N} then the scaled LDI-ladder modulator which is implemented in the prototype is shown in Fig. 4.16. The scaled coefficients ensure that all integrators will have maximum values in the same range as the first integrator, thereby maximizing the dynamic range of the signals throughout the ladder. The number of integer bits in the numerical representation is chosen so as to accommodate the maximum possible integrator value. The VHDL code describing the 6th order LDI-ladder modulator for OSR=32 described in Chapter 2. is given in Appendix B.

The state variables in an LDI resonator can also be scaled so as to make a most efficient use of the available register length. To do so we note that, when referring to Fig. 1.1, both integrators have the same gain at any given frequency of oscillation. In addition the product of the two integrator gains and of the loop coefficient k must equal 1, since the amplitude and frequency of oscillation is sustained over time. As a result each integrator contributes a gain equal to $1/\sqrt{k}$. The scaling is based on the smallest required non-zero

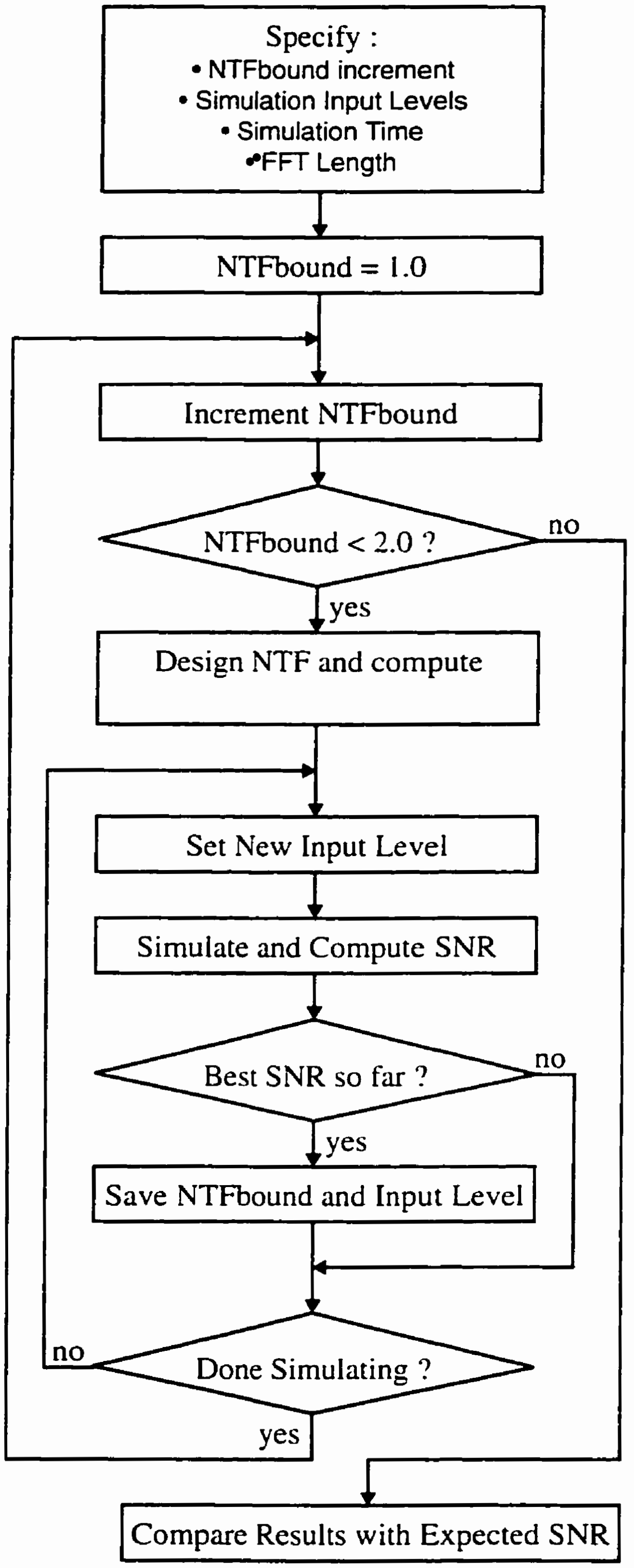


Fig. 4.14: Algorithm for designing a delta-sigma modulator with maximum SNR, for a given order and OSR.

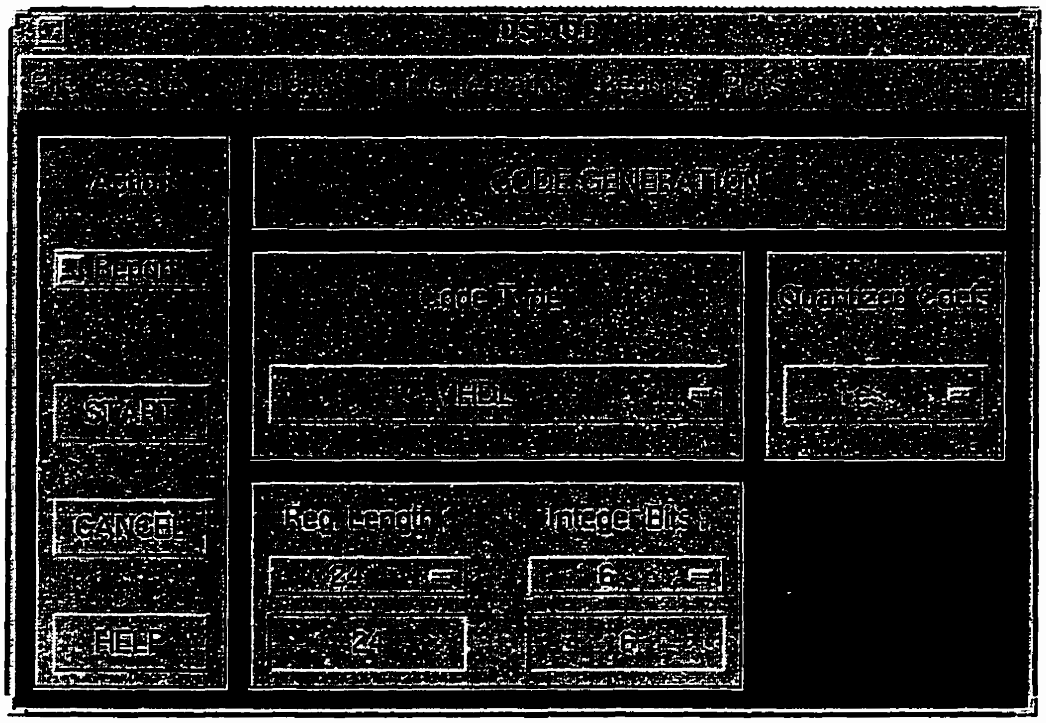


Fig. 4.15: Dialog box for prototype code generation.

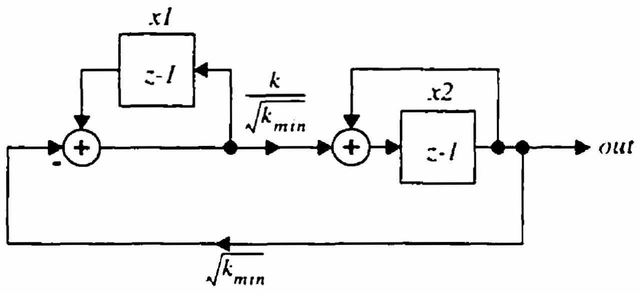


Fig. 4.17: Scaled Lossless Discrete Integrator (LDI) resonator when k<1.

value of k, called k_{min}, which determines the frequency resolution of the resonator. A properly scaled LDI resonator is then shown in Fig. 4.17 for the case in which k<1, and Fig. 4.17 for k>1. The same scaling operation is valid for a delta-sigma oscillator and is performed by DSMOD before VHDL code is generated for an oscillator prototype. Note that both k_0 and k_1 (as labelled in Fig. 3.1) must be scaled.

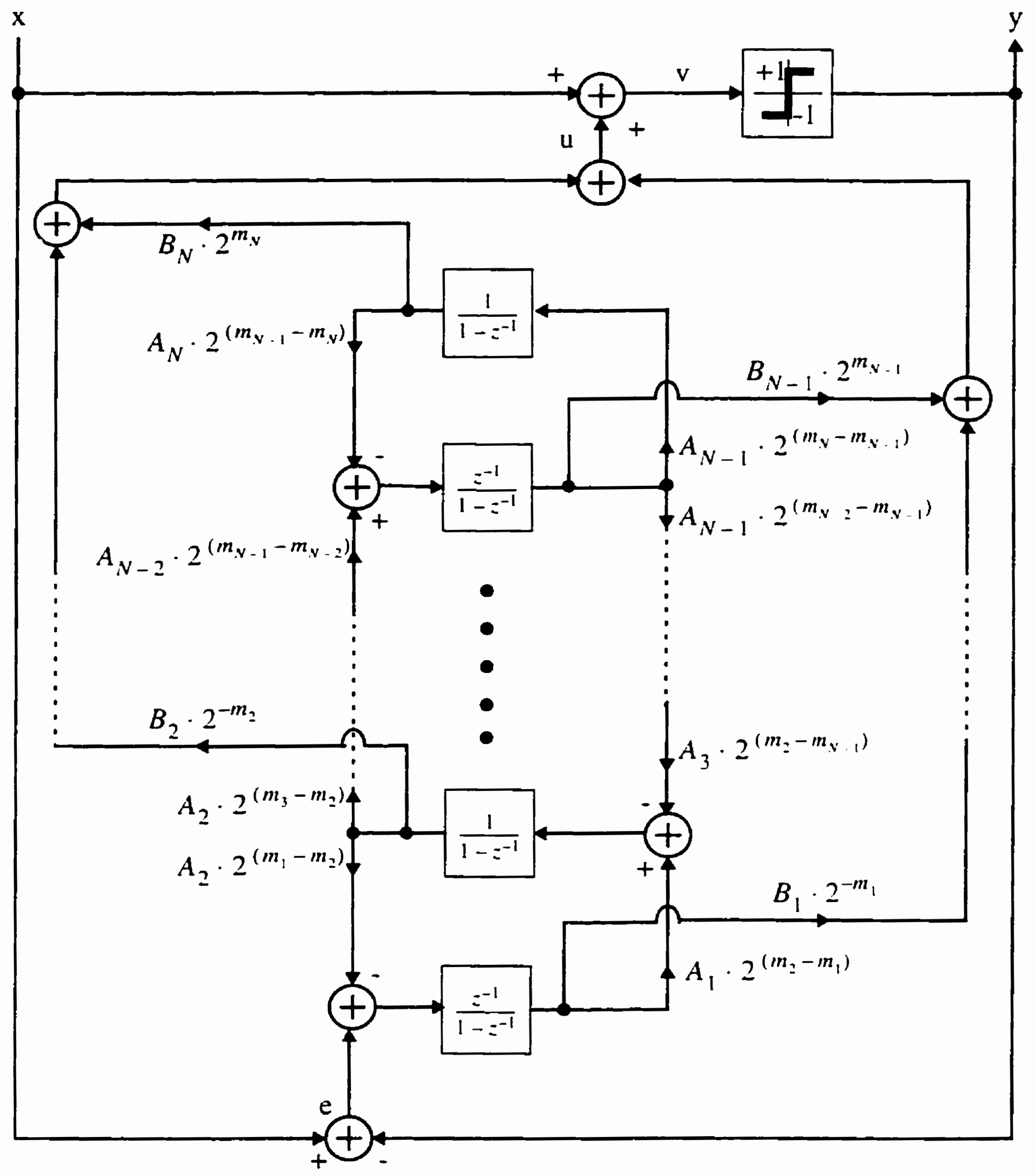

x
y
v
u
+1
-1
$B_N \cdot 2^{m_N}$
$A_N \cdot 2^{(m_{N-1} - m_N)}$
$B_{N-1} \cdot 2^{m_{N-1}}$
$A_{N-1} \cdot 2^{(m_N - m_{N-1})}$
$A_{N-1} \cdot 2^{(m_{N-2} - m_{N-1})}$
$A_{N-2} \cdot 2^{(m_{N-1} - m_{N-2})}$
$B_2 \cdot 2^{-m_2}$
$A_3 \cdot 2^{(m_2 - m_{N-1})}$
$A_2 \cdot 2^{(m_3 - m_2)}$
$A_2 \cdot 2^{(m_1 - m_2)}$
$B_1 \cdot 2^{-m_1}$
$A_1 \cdot 2^{(m_2 - m_1)}$
$\frac{1}{1-z^{-1}}$
$\frac{z^{-1}}{1-z^{-1}}$
e

Fig. 4.16: Scaled LDI-ladder-based modulator used to implement the prototypes.

4.5 Possible Improvements

This tool is still undergoing development. New features are added as research on signal generation progresses.

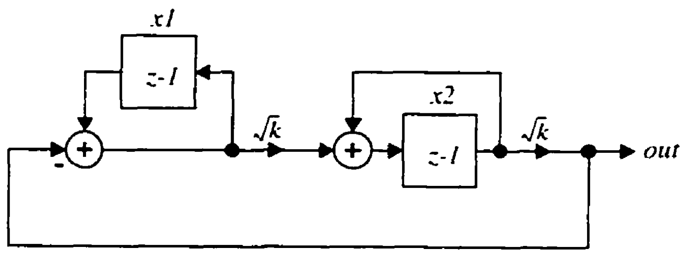


Fig. 4.18: Scaled Lossless Discrete Integrator (LDI) resonator when k>1.

4.5.1 Implementation

The computing speed of DSMOD could be greatly increased if it were written in a lower-level language, such as C. This comes about principally because DSMOD uses many iterative loops, such as in the coefficient-quantization modules, which are inefficiently run by the MATLAB software. Fortunately it may be possible to avoid reprogramming the tool from scratch by automatically generating C-code from the MATLAB code, using a newly released MATLAB product. This strategy would allow one to maintain MATLAB as the development platform for DSMOD, with all the ease-of-debugging offered by interactivity and interpreted code.

4.5.2 Improving Modulator Design

Arbitrary NTF shapes are greatly desirable in the context of mixed-signal testing applications, as argued in Section 3.5.3. A module similar to FiltorX could be developed for this purpose, or alternatively FiltorX [46] itself could be interfaced to DSMOD or even re-coded for MATLAB.

Because of its modularity, the tool can easily be extended to include new modulator and oscillator topologies, as well as new prototyping technologies. In particular, modulator topologies optimized for D/A and A/D conversion could be included in the tool. A module for scaling coefficients for SC implementation can be designed, as well as another one for decimation and anti-aliasing filters. The trend proposed here is to evolve toward a block-by-block system design tool, or collection of tools.

4.5.3 Improving Simulation

A module for fixed-point simulations would be quickly realizable based on the fixed-point module for SIMULINK, an extension package for MATLAB.

4.5.4 Improving Prototyping

For use in industrial settings, the tool could be interfaced with a silicon compiler via the VHDL language, so as to produce functional silicon layouts in a few hours. In this context, estimating the silicon area occupied by the oscillator or modulator would help with floor-planning an integrated circuit containing other components.

Finally, the size of each register and computational element in a given design could be computed. Simulations can help predict the largest possible numerical value at any given point in the circuit while a linear model of noise injection due to number truncation can predict the effect of finite-register length. The combination of these two techniques should allow one to choose the number of integer and fractional bits needed at every point in the circuit. This would further minimize the hardware cost of the designed circuits.

4.6 Conclusion

Most of the DSMOD software is written for MATLAB, a programmable, general-purpose matrix-algebra software. The advantages are multiple. First, the tool is based on a powerful, proven mathematical engine. Second, MATLAB provides an ideal flexible work-area to supplement the tool and to allow it to communicate with other similar tools. Lastly, the code and the user-interface are usable without any changes across all computer platforms supported by MATLAB. For these reasons this tool could easily incorporate contributions from widespread sources.

This chapter explained how and why our CAD tool is crucial in allowing a specific class of signal generator to be painlessly incorporated in more complex systems. The tool is based on a design module specialized for delta-sigma modulators and LDI resonators, and on simulation and prototyping modules for rapid and easy design validation. Our research group has used this tool very successfully in its own research on signal generation.

Such a tool is usable by system-level designers with little knowledge of delta-sigma modulation for designing self-testable system. It can be viewed as one among a collection of expert tools that could be used to quickly assemble complex systems at the VHDL-code level or signal-flow-graph level.

Chapter 5

Conclusions

5.1 Discussion of Results

The aim of the research presented herein has been to explore ways in which delta-sigma signal generation could be improved. Two types of improvements were sought: to increase the signal quality for a given bandwidth, and to design delta-sigma oscillators stable over an arbitrary signal band. Chapters 2 and 3 presented low-hardware cost solutions to these problems. One is the LDI-ladder-based delta-sigma modulator with single-bit output and unity STF, which can be designed to have cheaply-implementable power-of-two coefficients. Another is the stable delta-sigma oscillator topology, which makes use of an additional feedback loop as compared to the original oscillator design. It has been argued why these particular circuits were better suited for use in delta-sigma signal generation than other ones of the same kind. Multitone generation has been addressed too and shown to require very simple modifications to the single-tone circuits.

However this thesis is meant to be more than simply an account of these results. It strives to formalize the methods by which the circuits are designed. One key benefit of such a formalization is the possibility of automating these design methods. In fact, many of them are computationally intensive and necessitate automation. Another goal has been to include prototyping into a fast design cycle. Prototyping using Field-Programmable devices offers a valuable compromise between lengthy and imprecise simulations and costly silicon prototypes. Not only do the FPGA prototypes reported herein support the validity of our results, they also demonstrate how prototyping can be used to rapidly close the design cycle with a hardware-level validation of a given design.

As a result of the needs for automation and prototyping, DSMOD has been developed into a full-fledged delta-sigma oscillator design software, and presented in Chapter 4. It demonstrates the following points: that it is possible to simultaneously develop new circuits and the tools needed to design and evaluate them; and that a complex computer-aided design tool for a very specific class of digital circuits can be built on the MATLAB platform and used to design working prototypes in but a few hours, with no expert knowledge required.

The motivation for this research is the need for low-cost self-test solutions for mixed-signal circuits and systems. The fact that delta-sigma oscillators are digital circuits makes them suitable for a system-design strategy in which hardware is re-used to implement the self-test functionality. This should contribute to lowering the cost of endowing a given mixed-signal device or system with self-test capability. Perhaps the greatest advantage of these circuits is that they are fully programmable and are as insensitive to process and temperature variations as any digital hardware; the same is certainly not true of signal generation circuits based on analog solutions.

5.2 Future Directions

The original 2^{nd}-order oscillator has been successfully incorporated in a voiceband codec [2][3]. The novel oscillators presented here should make possible the implementation of self-testable mixed-signal circuits more complex and more demanding in performance. Telecommunications systems, with their strict constraints on reliability and the huge costs associated with their maintenance, should benefit greatly from on-board high-precision testing capability.

Although the issue of signal generation has now been addressed in much depth, the mixed-signal self-test algorithms and their implementation have not been greatly debated since [3] was published. There is a definite need to upgrade these algorithms to make use of the new possibilities offered by arbitrary-precision delta-sigma oscillators.

There also is a need to assess the cost trade-offs involved in implementing self-testability in mixed-signal integrated circuits, boards and systems. Although such issues are ultimately dealt with by industry, pointing to specific, economically viable applications of mixed-signal self-test would allow industry to more rapidly take over the burden of developing prototypes -- and working products.

Appendix A
Windowing

A.1 Principles of Windowing

Consider the task of testing a linear time-invariant discrete-time system. Since the system is L.T.I., one can obtain its transfer function (frequency response) and predict the Fourier transform of the output given the input, using Fourier analysis. If the system's transfer function is $H(j\omega)$ and the input is $x(n)$, then the Fourier transform of the ouput $y(n)$ is given by (see [39], p. 203):

$$Y(j\omega) = H(j\omega) \sum_{n=-\infty}^{\infty} x(n) e^{j\omega n}. \qquad \text{(A.1)}$$

In fact, information about the expected output signal is not only easily obtained but also easily expressed in the frequency domain. For this reason the specifications of a linear system are most often given in the frequency domain, and the goal of testing is then to decide whether the frequency domain specifications are met.

The design is usually tested against its specifications either through simulation or by fabricating and testing a real implementation. In both cases, typical tests in the frequency domain consist in observing the system's response to a sinusoidal or DC input and comparing it to the theoretical response at the same frequency.

One major problem arises here: sinusoidal and DC inputs extend over an infinite period of time, and so do their corresponding outputs. Simulations and physical tests, on the contrary, can generate only a finite number of output samples. The complete output of the real system is thus unavailable, and it is impossible to directly compare the Fourier transform of the desired output to the one of the actual output.

However, the finite number of output samples can be processed so that they give rise to a good approximation of the Fourier transform of the infinite-duration output (which would be obtained if the test or the simulation ran forever). Windowing is the name given to this type of processing, in which N output samples are weighted according to their time-index, so as to reduce the effects of using a finite-extent output sequence.

In the mathematical formulation that follows, it is assumed that the complete output sequence $y(n)$ is available, and that the window has zero value everywhere except in the observation interval. This trick allows us to think of the windowed, finite-duration output as the product of the complete output with another sequence, even though the complete output is unavailable. Additional assumptions are that N is even and that the observation interval is $[-N/2, N/2-1]$.

In addition to being equal to zero everywhere except in the observation interval, the window $w(n)$ must be even except for its last value in the observation interval, which equals 0 (see [48]). To summarize, the constraints on $w(n)$ are:

$$w(n) = 0 \qquad |n| > \frac{N}{2}. \tag{A.2}$$

$$w(n) = w(-n) \qquad n \neq \frac{N}{2}. \tag{A.3}$$

$$w\left(\frac{N}{2}\right) = 0. \tag{A.4}$$

The Fourier transform of the windowed output is given by:

$$Y_W(j\omega) = \sum_{n=-\infty}^{\infty} y(n)\, w(n)\, e^{j\omega n}. \tag{A.5}$$

Before examining in details the effects of windowing, it is pertinent to realize that nothing tells us *a-priori* that the Fourier transform of a finite number of unprocessed samples will appropriately approximate the Fourier transform of the complete signal. In fact the key to windowing is choosing how to weight the samples in the finite-duration obser-

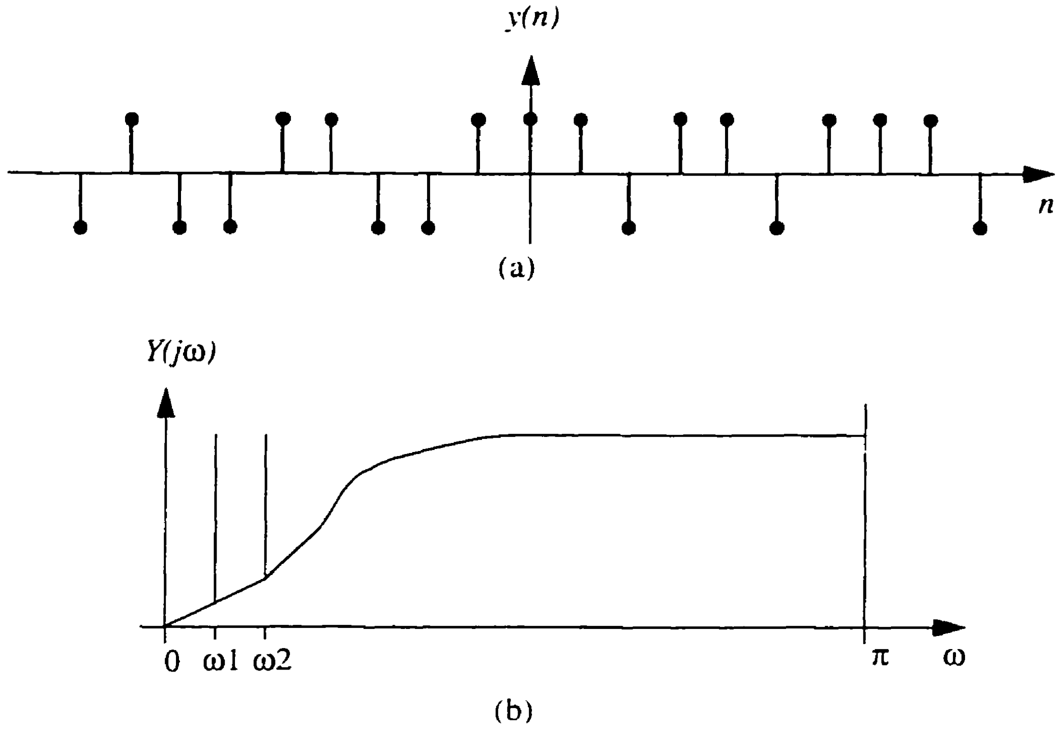


Fig. A.1: (a) A typical 1-bit noise-shaped data converter output sequence; (b) its Fourier transform

vation interval so that their Fourier transform constitutes a good approximation, depending on the particularities of the system [48] and the features of the output signal which are to be observed.

The specific effects of windowing are best explained through an example. Fig. A.1 shows part of one possible output of a discrete-time system, namely the output of a one-bit delta-sigma converter. The Fourier transform of the output is also shown. (In the case of the delta-sigma converter, which is not a linear system, the expected Fourier transform of the output is obtained after the system has been linearized.) Note that the output sequence has infinite duration. In this particular example the Fourier transform indicates that the signal is composed of two low-frequency sinusoids plus a fair amount of noise at high-frequency and very little noise at low-frequency

As previsously explained, only a finite-extent output sequence is obtained from simulation or testing, and this truncated output can be thought of as the product of the infi-

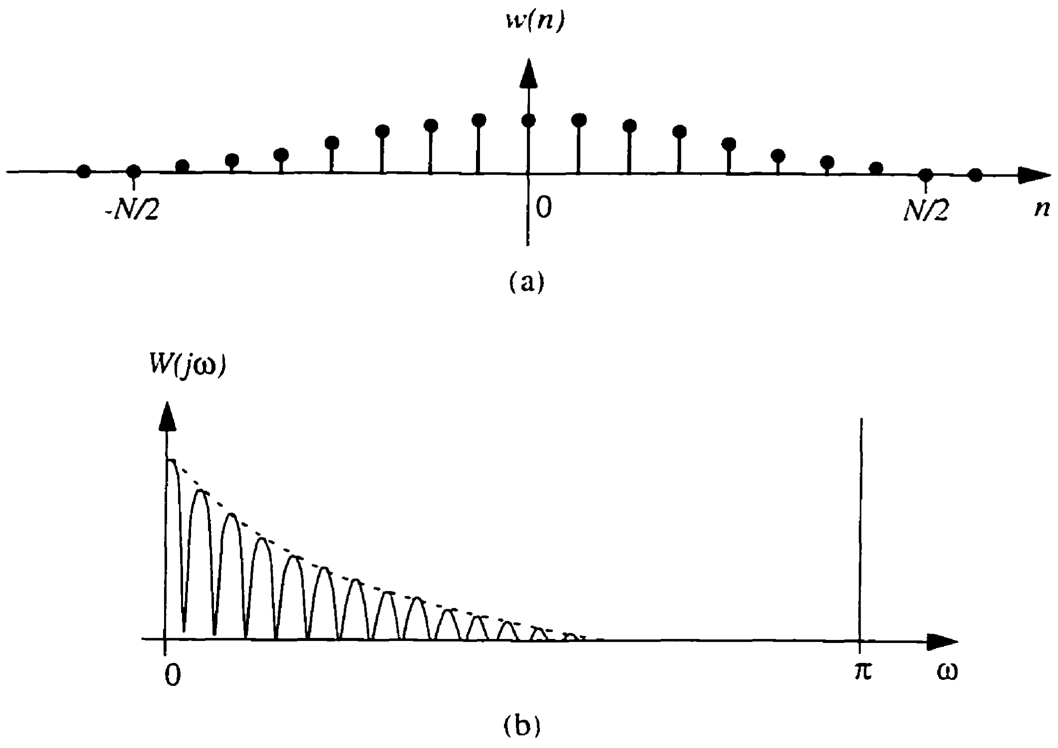


Fig. A.2: (a) Processing window; (b) Fourier transform of the same window.

nite-duration sequence with a window which has non-zero values only in the observation interval. A typical window and its Fourier transform are shown in Fig. A.3.

Multiplication of two signals in the time-domain becomes convolution in the frequency domain. Thus the Fourier transform of the infinite-duration signal (Fig. A.1(b)) is convolved with the Fourier transform of the window (Fig. A.3(b)) to result in the Fourier transform of the windowed, finite-duration signal, shown in Fig. A.3.

Windowing results in a deformation of the initial Fourier transform. Tones at single frequencies are spread over a non-zero frequency interval; the information at frequencies between two nearby tones is thus lost. Furthermore, the low-power noise floor at low frequency is lost under the smearing of the more powerful tones and high-frequency noise.

In spite of seemingly disastrous results, a carefully chosen window does preserve the essential features of the Fourier transform. If the initial signal contains tones at very nearby frequencies, then a window with a narrow mainlobe must be used. If the dynamic range is initially very large, then a window with low sidelobes is to be chosen. If, as in the

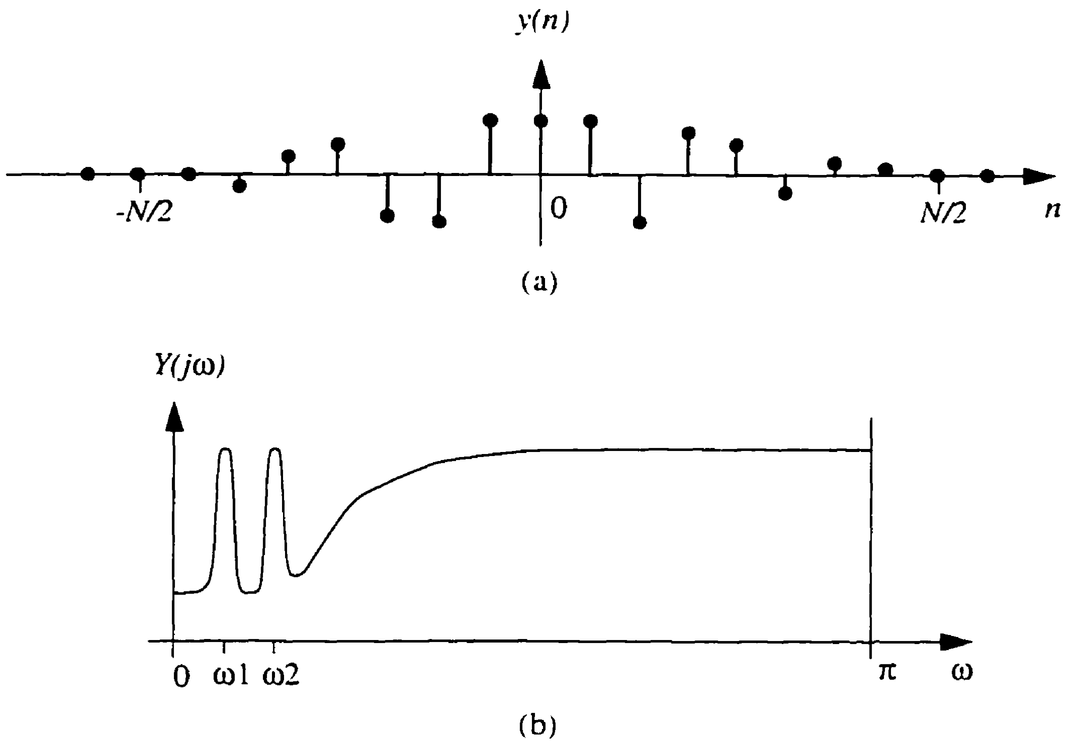


Fig. A.3: (a) Windowed output sequence; (b) Fourier transform of the windowed sequence.

case of noise-shaped signals, there are two frequency regions with very different power levels, then a window with a fast sidelobe decay is preferred so as to minimize the interference of one region over the other. The theoretical details of windowing are presented in [48].

Finally, the Fourier transform of the windowed signal is computed at discrete frequencies by an FFT algorithm; in the time domain this corresponds to periodically extending the signal, as shown in Fig. A.3.

Additional considerations come into play in obtaining a reasonable approximation to the desired Fourier transform.

First, the effect of start-up transients must be minimized. This is accomplished by discarding the initial samples of a simulation or a test, and keeping the rest of the samples for analysis. As a rule of thumb, the second half of the results can be used for frequency analysis, while the first half should be ignored.

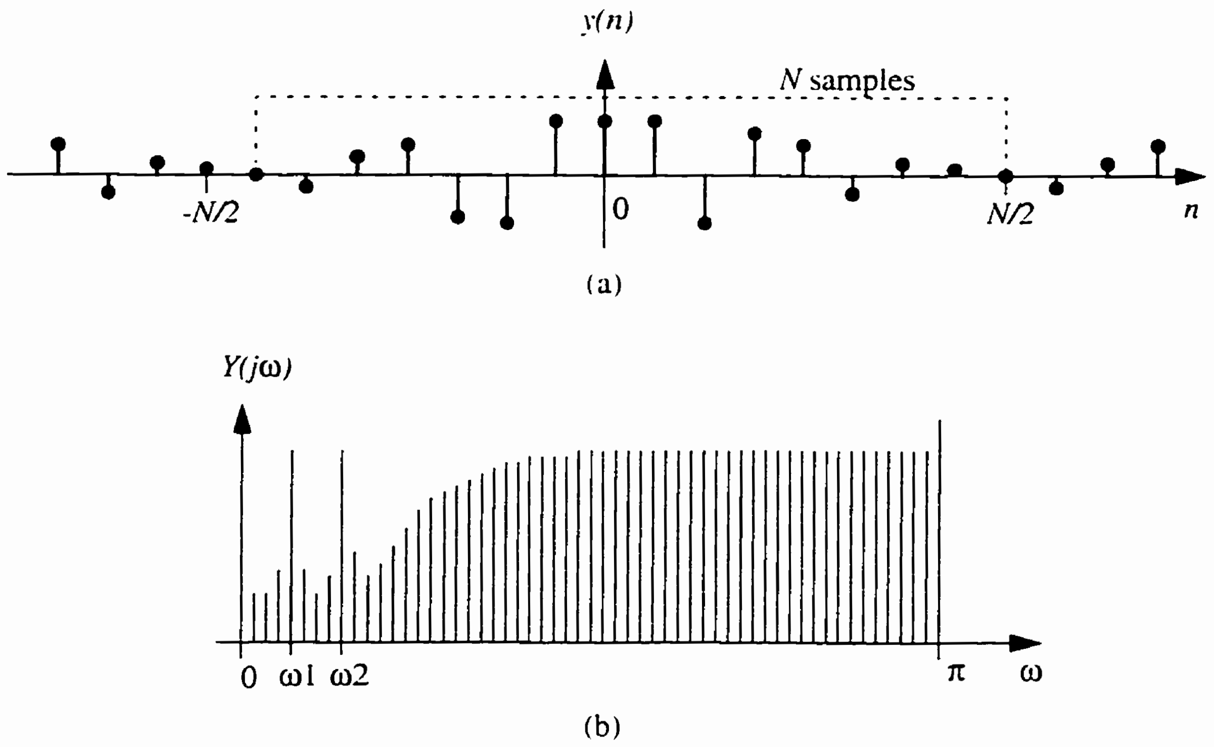


Fig. A.4: (a) Periodic extension of the finite-duration sequence; (b) discrete Fourier transform of the finite-duration sequence.

The second consideration is a trick that takes advantage of a property of most windows and of the use of the discrete Fourier transform. The Fourier transform of an N-point window equals zero at all multiples of $2\pi/N$ except at low-frequencies. Also, the discrete Fourier transform has values only at frequencies that are multiples of $2\pi/N$. Thus, a tone at one of these discretized frequencies will spread only to nearby frequency bins. In practical terms, the period of such a tone must be a divisor of the sample-size.

A.2 High-Performance Windows

Nuttall [45] identifies a basic tradeoff in designing or choosing a window: that between a low main sidelobe level and fast-decaying sidelobes at higher frequencies. Here we focus on windows providing fast-decaying sidelobes, since we are worried about very-high resolution single tones being smeared across the signal band because of their inevitable incoherence due to delta-sigma modulation.

Fast decaying sidelobes are obtained when many derivatives of the window at its edges are continuous. In general, a raised-cosine window is defined as:

$$w(n) = \sum_{k=0}^{K} a_k \cos\frac{2\pi n}{N}. \qquad \text{(A.6)}$$

where $K+1$ is the number of cosine terms making up the window and N is the window length. A $K+1$-term window can be made to have all its derivatives up to the $(2K-1)^{th}$ equal to zero everywhere, provided the following equation is respected:

$$\begin{bmatrix} 1 & 1 & \dots & 1 & \dots & 1 \\ 1 & 1 & \dots & (-1)^k & \dots & (-1)^K \\ 0 & 1 & \dots & (-1)^k k^2 & \dots & (-1)^K K^2 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 1 & \dots & (-1)^k k^{2l} & \dots & (-1)^K K^{2l} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 1 & \dots & (-1)^k k^{2K} & \dots & (-1)^K K^{2K} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \dots \\ a_k \\ \dots \\ a_K \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ \dots \\ 0 \\ \dots \\ 0 \end{bmatrix}. \qquad \text{(A.7)}$$

As an example, the coefficients of the 4-term, continuous-fifth-derivative raised-cosine window are given in [45] to be:

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 10/32 \\ 15/32 \\ 6/32 \\ 1/32 \end{bmatrix}. \qquad \text{(A.8)}$$

Appendix B

VHDL Source Code

B.1 VHDL code for 6th-Order Modulator, OSR=32

The following code describes a 6th-order modulator designed by DSMOD as described in Section 4.4.8. This code was entirely generated by DSMOD. Note that only the higher level of hierarchy describing the design is given here. The code describing the integrators, the adders and the registers is not shown.

```
-----------------------------------------------------------------
-- Delta-Sigma Modulator
-- Generated by DSMOD
--    Xavier Haurie & Gordon W. Roberts
--    MACS Laboratory, McGill University (EE Dept.)
--    Montreal (QC) Canada
--    email: dsmod@macs.ee.mcgill.ca
-- Modulator Parameters:
--    Structure: Scaled Ladder
--    Passband:  any (LP, BP or HP)
--    Order:     6
-----------------------------------------------------------------

library synopsys;
use synopsys.bv_arithmetic.all;
use work.arithmetic.all;
use work.sequential.all;
use work.blocks.all;

entity modulator is
  generic (size: positive := 1;
           int_bits: positive := 1);
  port (x: out bit;
       a: in bit_vector(size-1 downto 0);
       clk: in bit;
```

```
        reset: in bit);
  -- pragma template
end modulator;

architecture structural of modulator is

  signal subfirst_out: bit_vector(size-1 downto 0);
  signal sub1_out: bit_vector(size-1 downto 0);
  signal int1_out: bit_vector(size-1 downto 0);
  signal addb_in1: bit_vector(size+2-1 downto 0);
  signal sub2_out: bit_vector(size-1 downto 0);
  signal int2_out: bit_vector(size-1 downto 0);
  signal add2_out: bit_vector(size-1 downto 0);
  signal addb_in2: bit_vector(size+2-1 downto 0);
  signal sub3_out: bit_vector(size-1 downto 0);
  signal int3_out: bit_vector(size-1 downto 0);
  signal add3_out: bit_vector(size-1 downto 0);
  signal addb_in3: bit_vector(size+2-1 downto 0);
  signal sub4_out: bit_vector(size-1 downto 0);
  signal int4_out: bit_vector(size-1 downto 0);
  signal add4_out: bit_vector(size-1 downto 0);
  signal addb_in4: bit_vector(size+2-1 downto 0);
  signal sub5_out: bit_vector(size-1 downto 0);
  signal int5_out: bit_vector(size-1 downto 0);
  signal add5_out: bit_vector(size-1 downto 0);
  signal addb_in5: bit_vector(size+2-1 downto 0);
  signal addb_in6: bit_vector(size+2-1 downto 0);
  signal int6_out: bit_vector(size-1 downto 0);
  signal addb_out: bit_vector(size+2-1 downto 0);
  signal addq_out: bit_vector(size+2-1 downto 0);
  signal addq_in1: bit_vector(size+2-1 downto 0);
  signal comp_out: bit_vector(size-1 downto 0);
  signal mod_in: bit_vector(size-1 downto 0);
  signal a1u_out,a1u_in: bit_vector(size-1 downto 0);
  signal a1u_1_out,a1u_1_in: bit_vector(size-1 downto 0);
  signal a2u_out,a2u_in: bit_vector(size-1 downto 0);
  signal a2u_1_out,a2u_1_in: bit_vector(size-1 downto 0);
  signal a3u_out,a3u_in: bit_vector(size-1 downto 0);
  signal a3u_1_out,a3u_1_in: bit_vector(size-1 downto 0);
  signal a4u_out,a4u_in: bit_vector(size-1 downto 0);
  signal a4u_1_out,a4u_1_in: bit_vector(size-1 downto 0);
  signal a5u_out,a5u_in: bit_vector(size-1 downto 0);
  signal a5u_1_out,a5u_1_in: bit_vector(size-1 downto 0);
  signal a2d_out,a2d_in: bit_vector(size-1 downto 0);
  signal a2d_1_out,a2d_1_in: bit_vector(size-1 downto 0);
  signal a3d_out,a3d_in: bit_vector(size-1 downto 0);
  signal a3d_1_out,a3d_1_in: bit_vector(size-1 downto 0);
  signal a4d_out,a4d_in: bit_vector(size-1 downto 0);
  signal a4d_1_out,a4d_1_in: bit_vector(size-1 downto 0);
  signal a5d_out,a5d_in: bit_vector(size-1 downto 0);
  signal a5d_1_out,a5d_1_in: bit_vector(size-1 downto 0);
  signal a6d_out,a6d_in: bit_vector(size-1 downto 0);
  signal a6d_1_out,a6d_1_in: bit_vector(size-1 downto 0);
  signal b1_out,b1_in: bit_vector(size-1 downto 0);
  signal b1_1_out,b1_1_in: bit_vector(size-1 downto 0);
  signal b2_out,b2_in: bit_vector(size-1 downto 0);
  signal b2_1_out,b2_1_in: bit_vector(size-1 downto 0);
  signal b3_out,b3_in: bit_vector(size-1 downto 0);
  signal b3_1_out,b3_1_in: bit_vector(size-1 downto 0);
  signal b4_out,b4_in: bit_vector(size-1 downto 0);
  signal b4_1_out,b4_1_in: bit_vector(size-1 downto 0);
  signal b5_out,b5_in: bit_vector(size-1 downto 0);
  signal b5_1_out,b5_1_in: bit_vector(size-1 downto 0);
  signal b6_out,b6_in: bit_vector(size-1 downto 0);
  signal b6_1_out,b6_1_in: bit_vector(size-1 downto 0);
begin

  -- Connect 1-bit output to comparator multi-bit output
  mod_in <= a(size-1 downto 0);

  -- Connect 1-bit output to comparator multi-bit output
  x <= not comp_out(size-1);

  -- instantiate comparator
  comp: comparator
    generic map(size, int_bits)
    port map(comp_out, addq_out(size+2-1));

  addq: adder2
    generic map(size+2,0,0)
    port map(addq_out,addq_in1,addb_out);
  addq_in1 <= sxt(mod_in,size+2);

  addb: adder6
    generic map(size+2,0,0,0,0,0,0)
    port
map(addb_out,addb_in1,addb_in2,addb_in3,addb_in4,addb_in5,addb_in6);
  addb_in1 <= sxt(b1_out,size+2);
  addb_in2 <= sxt(b2_out,size+2);
  addb_in3 <= sxt(b3_out,size+2);
  addb_in4 <= sxt(b4_out,size+2);
  addb_in5 <= sxt(b5_out,size+2);
  addb_in6 <= sxt(b6_out,size+2);

  subfirst: adder2
    generic map(size,0,1)
    port map(subfirst_out,mod_in,comp_out);

  sub1: adder2
    generic map(size,0,1)
    port map(sub1_out,subfirst_out,a2d_out);

  sub2: adder2
    generic map(size,0,1)
    port map(sub2_out,a1u_out,a3d_out);

  sub3: adder2
    generic map(size,0,1)
    port map(sub3_out,a2u_out,a4d_out);

  sub4: adder2
    generic map(size,0,1)
    port map(sub4_out,a3u_out,a5d_out);

  sub5: adder2
    generic map(size,0,1)
    port map(sub5_out,a4u_out,a6d_out);

  int1: b_int
    generic map(size)
    port map(int1_out,sub1_out,clk,reset);

  int2: f_int
    generic map(size)
    port map(int2_out,sub2_out,clk,reset);

  int3: b_int
    generic map(size)
    port map(int3_out,sub3_out,clk,reset);

  int4: f_int
    generic map(size)
    port map(int4_out,sub4_out,clk,reset);

  int5: b_int
    generic map(size)
    port map(int5_out,sub5_out,clk,reset);

  int6: f_int
    generic map(size)
    port map(int6_out,a5u_out,clk,reset);

  a1u_in <= int1_out;
  a1u: shift_right
    generic map(size,1)
    port map(a1u_out,a1u_in);

  a2u_in <= int2_out;
  a2u: shift_right
    generic map(size,3)
    port map(a2u_out,a2u_in);

  a3u_in <= int3_out;
  a3u: shift_right
    generic map(size,2)
    port map(a3u_out,a3u_in);

  a4u_in <= int4_out;
  a4u: shift_right
    generic map(size,3)
    port map(a4u_out,a4u_in);
```

```
a5u_in <= int5_out;
a5u: shift_right
  generic map(size,4)
  port map(a5u_out,a5u_in);

a2d_in <= int2_out;
a2d: shift_right
  generic map(size,8)
  port map(a2d_out,a2d_in);

a3d_in <= int3_out;
a3d: shift_right
  generic map(size,5)
  port map(a3d_out,a3d_in);

a4d_in <= int4_out;
a4d: shift_right
  generic map(size,7)
  port map(a4d_out,a4d_in);

a5d_in <= int5_out;
a5d: shift_right
  generic map(size,6)
  port map(a5d_out,a5d_in);

a6d_in <= int6_out;
a6d: shift_right
  generic map(size,4)
  port map(a6d_out,a6d_in);

b1_in <= int1_out;
b1: shift_right
  generic map(size,1)
  port map(b1_out,b1_in);

b2_in <= int2_out;
b2: shift_right
  generic map(size,1)
  port map(b2_out,b2_in);

b3_in <= int3_out;
b3: shift_right
  generic map(size,1)
  port map(b3_out,b3_in);

b4_in <= int4_out;
b4: shift_right
  generic map(size,1)
  port map(b4_out,b4_in);

b5_in <= int5_out;
b5: shift_right
  generic map(size,3)
  port map(b5_out,b5_in);
  b6_in <= int6_out;
  b6: shift_right
    generic map(size,3)
    port map(b6_out,b6_in);

end structural;
```

Bibliography

[1] P. H. Bardell, W. H. McAnney and J. Savir, *Buit-In Test for VLSI*. Wiley Interscience, New York, 1987.

[2] M. F. Toner and G. W. Roberts, " A BIST Scheme for an SNR Test of a Sigma-Delta ADC," *IEEE International Test Conference*, Baltimore, Maryland, pp. 805-814, Oct. 1993.

[3] M. F. Toner and G.W. Roberts, "A BIST Scheme for an SNR, Gain Tracking, and Frequency Response Test of a Sigma-Delta ADC," *IEEE Trans. on Circuits and Systems -- II: Analog and Digital Signal Processing*, Vol. 41, No. 12, pp. 1-15, Jan. 1995.

[4] S. Sunter, "The P1149.4 Mixed Signal Test Bus: Costs and Benefits", *IEEE International Test Conference*, Washington, D.C., pp. 444-450, Oct. 1995.

[5] A. K. Lu, G. W. Roberts and D. Johns, "A high-quality analog oscillator using oversampling D/A conversion techniques," *IEEE Trans. on Circuits and Systems -- II: Analog and Digital Signal Processing*, Vol. 41, No. 7, pp. 437-444, July 1994.

[6] A. K. Lu and G. W. Roberts, "An Oversampled-Based Analog Multi-Tone Signal Generator", *Proceedings IEEE International Test Conference*, Washington D.C., pp. 650-659, Oct. 1994.

[7] A. K. Lu and G. W. Roberts, "An Oversampling-Based Analog Multi-Tone Signal Generator," *IEEE Trans. on Circuits and Systems -- II: Analog and Digital Signal Processing*, accepted, April 1995.

[8] G. W. Roberts and A. K. Lu, *Analog Signal Generation For Built-In Self-Test Of Mixed-Signal Integrated Circuits*, Kluwer Academic Publishers, Norwell, MA, USA, 1994 (120 pages).

[9] A. S. Sedra and K. C. Smith, *Microelectronics Circuits*, 3rd ed., HRW-Saunders, Florida, 1991.

[10] H. T. Nicholas and H. Samueli, "A 150MHz Direct Digital Frequency Synthesizer in 1.25 mm CMOS with -90 dBc Spurious Performance", *IEEE Journal of Solid-State Circuits*, vol. 26, pp. 1959-1969, Dec. 1991.

[11] S. D. Worthington, *Limit Cycles and Sinusoidal Oscillations in Digital Systems*, Master's Thesis, University of Calgary, Alberta, September 1992.

[12] B. R. Veillette and G. W. Roberts. "High-Frequency Sinusoidal Generation Using Delta-Sigma Modulation Techniques". *Proc. IEEE International Symposium on Circuits and Systems*. pp. 637-640. May 1995.

[13] B. R. Veillette and G. W. Roberts. "A Built-In Self-Test Strategy for Wireless Communication Systems," *Proceedings IEEE International Test Conference*. Washington D.C.. October 1995.

[14] B. R. Veillette and G. W. Roberts. "FM Signal Generation Using Delta-Sigma Oscillators." *Proceedings IEEE International Symposium on Circuits and Systems*. Atlanta. Georgia. May 1996.

[15] B. R. Veillette. *A Study of Delta-Sigma Oscillator Circuits*. Master's Thesis. McGill University. August 1995.

[16] X. Haurie and G. W. Roberts. "Arbitrary-Precision Signal Generation for Bandlimited Mixed-Signal Testing," *Proceedings IEEE International Test Conference*. Washington D.C.. October 1995.

[17] X. Haurie and G. W. Roberts. "A Multiplier-Free Structure for 1-Bit Digital Delta-Sigma Modulators". Presented at the *Midwest Symposium on Circuits and Systems*. August 1995.

[18] X. Haurie and G. W. Roberts. "A Design. Simulation and Synthesis Tool for Delta-Sigma-Modulator-Based Signal Sources". *International Symposium on Circuits and Systems*. Atlanta GA. May 1996.

[19] M. W. Hauser. "Principles of Oversampling A/D Conversion". *J. Audio Eng. Soc.*. vol. 39. no. 1/2. pp. 3-26. January/February 1991.

[20] P. M. Aziz. H. V. Sorensen and J. V. Der Spiegel. "An Overview of Sigma-Delta Converters". *IEEE Signal Processing Magazine*. pp. 61-84. January 1996.

[21] J. C. Candy and G. C. Temes editors. *Oversampling Delta-Sigma Data Converters*. IEEE Press. 1992.

[22] G. R. Ritchie. J. C. Candy and W. H. Ninke. "Interpolative digital-to-analog converters". *IEEE Trans. Commun.*. vol. COM-22. pp. 1797-1806. November 1974.

[23] P. J. A. Naus. E. C. Dijkmans. E. F. Stikvoort. A. J. McKnight. D. J. Holland and W. Brandinal. "A CMOS Stereo 16-Bit D/A Converter for Digital Audio". *J. Solid State Circuits*. vol. SC-22. no. 3. pp. 390-395. June 1987.

[24] L. Risbo. "FPGA Based 32 Times Oversampling 8th-Order Sigma-Delta Audio DAC". presented at the *96th Convention of the Audio Eng. Soc.*. February 1994.

[25] R. Schreier, "An Emprirical Study of Higher-Order Single-Bit Delta-Sigma Modulators", *IEEE Trans. Circuits Syst.*, vol. CAS-40, no. 8, pp. 461-466, August 1993.

[26] L. R. Carley and J. Kenney, "A 16-Bit 4'th Order Noise-Shaping D/A Converter", *IEEE Proc. CICC*, pp. 21.7.1-21.7.4, 1988.

[27] Y. Matsuya, K. Uchimura, A. Iwata, T. Kobayashi, M. Ishikawa and T. Yoshitome, "A 16-Bit Oversampling A/D Conversion Technology Using Triple Integration Noise Shaping", *IEEE J. Solid State Circuits*, vol. SC-22, no. 6, pp. 921-929, December 1987.

[28] R. Khoini-Poorfard and D. A. Johns, "Mismatch Effects in Time-Interleaved Oversampling Converters", in *Proc. IEEE Int. Symp. Circuits Syst.*, pp. 429-432, May 1994.

[29] H. T. Jensen and I. Galton, "A Robust Parallel Delta-Sigma A/D converter Architecture", *Proc. IEEE Int. Symp. Circuits Syst.*, pp. 1340-1343, May 1995.

[30] I. Galton, "Noise Shaping D/A Converters for $\Delta\Sigma$ Modulation", *Proc. IEEE Int. Symp. Circuits Syst.*, pp. 441-443, May 1996.

[31] R. Schreier and M. Snelgrove, "Bandpass Sigma-Delta Modulation", *Electronics Letters*, pp. 1560-1561, November 1989.

[32] S. Jantzi, R. Schreier and M. Snelgrove, "Complex Bandpass Sigma-Delta Converter for Digital Radio", *Proc. IEEE Int. Symp. Circuits and Systems*, pp. 453-456, 1994.

[33] R. Schreier, *Noise-Shaped Coding*, Ph. D. Thesis, University of Toronto, 1991.

[34] W. L. Lee, *A Novel Higher Order Interpolative Modulator Topology for High Resolution Oversampling A/D Converters*, Master's Thesis, Massachusetts Institute of Technology, June 1987.

[35] K. C. Chao, S. Nadeem, W. L. Lee, "A Higher Order Topology for Interpolative Modulators for Oversampling A/D Converters", *IEEE Trans. Circuits Syst. II*, vol. CAS-37, no. 3, pp. 309-318, March 1990.

[36] R.W Adams, P. F. Ferguson JR., A. Ganesan, S. Vincelette, A. Volpe and R. Libert, "Theory and Practical Implementation of a Fifth-Order Sigma-Delta A/D Converter", *J. Audio Eng. Soc.*, vol. 39, no. 7/8, pp. 515-528, July/August 1991.

[37] D. R. Welland, B. P. Del Signore and E. J. Swanson, "A Stereo 16-Bit Delta-Sigma A/D Converter for Digital Audio", *J. Audio Eng. Soc.*, vol. 37, no. 6, pp. 476-486, June 1989.

[38] L. E. Turner, E. S. K. Liu and L. T. Bruton, "Digital LDI Ladder Design Using the Bilinear Transformation", in *Proc. IEEE Int. Symp. Circuits Syst.*, pp. 1017-1020, May 1984.

[39] J. G. Proakis and D. G. Manolakis, *Digital Signal Processing; Principles, Algorithms and Applications*, 2nd. ed., Macmillan, New-York, 1992.

[40] H. Samueli, "An Improved Search Algorithm for the Design of Multiplierless FIR filters with Powers-of-Two Coefficients", *IEEE Trans. Circuits Syst.*, vol. CAS-36, no. 7, pp. 1044-1047, July 1989.

[41] Q. Zhao and Y. Tadokoro, "A Simple Design of FIR Filters with Poers-of-Two Coefficients", *IEEE Trans. Circuits Syst. II*, vol. CAS-35, no. 5, May 1988.

[42] B. Zeng and Y. Neuvo, 'Design of Direct-Form FIR Digital Filters with Discrete Valued Coefficients", *Proc. IEEE Int. Symp. Circuits Syst.*, pp. 484-487, June 1991.

[43] T. Saramaki, "A Systematic Technique for Designing Highly Selective Multiplier-Free FIR Filters", *Proc. IEEE Int. Symp. Circuits Syst.*, pp. 484-487, June 1991.

[44] *Matlab Reference Guide*, The MathWorks Inc., Natick MA, August 1992.

[45] A. H. Nuttall, "Some Windows with Very Good Sidelobe Behavior", *IEEE Trans. Acc. Speech Sig. Proc*, Vol. ASSP-29, No. 1, February 1981.

[46] S. Jantzi, C. Ouslis and A. Sedra, "Transfer Function Design for $\Delta\Sigma$ Converters", in *Proc. IEEE Int. Symp. Circtuis Syst.*, pp. 433-436, May 1994.

[47] L. Risbo, "Stability Predictions for High-Order $\Sigma-\Delta$ Modulators Based on Quasi-linear Modeling", *Proc. IEEE Int. Symp. Circuits Syst.*, pp. 361-364, May 1994..

[48] F. J. Harris, "On the Use of Windows for Harmonic Analysis with the Discrete Fourier Transform", *Proceedings of the IEEE*, Vol. 66, No. 1, January 1978.

IMAGE EVALUATION TEST TARGET (QA-3)