VLSI IMPLEMENTATION OF A HIGH-SPEED DELTA-SIGMA ANALOG TO DIGITAL CONVERTER

A Thesis Presented to
The Faculty of the Russ College of Engineering and Technology
Ohio University

In Partial Fulfillment
of the Requirement for the Degree
Master of Science

By
Zheng Chen
November, 1997

THIS THESIS ENTITLED

VLSI IMPLEMENTATION OF A HIGH-SPEED

DELTA-SIGMA ANALOG TO DIGITAL CONVERTER

by Zheng Chen

has been approved

for the School of Electrical Engineering and Computer Science

and the Russ College of Engineering and Technology

ACKNOWLEDGEMENTS

I sincerely thank my thesis advisor, Dr. Janusz Starzyk, for his many hours of generous assistance. Not only did Dr. Starzyk recommend many useful references, but also attributed his insights in many theoretical issues. I will remember the refreshed feeling that I got each time I discussed with him a headache problem disturbing me for several days. That feeling gave me new energy and guided me to the right direction.

Special thanks go to my family and my dear friend, Ling Zhu, as well as his parents, whose support gave me spiritual strength in pursuing my ambition and my staying in the United States.

I am very thankful also, to Dr. Voula Georgopoulos, Dr. Xuefeng Fang at IDT Inc., and Mr. David Zar at Washington University in St. Louis, for their help on digital filter, switched-capacitor circuit, and MOSIS design kits, respectively. Finally, thanks to all my friends, who assisted me upon request and were very helpful in their suggestions.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

The emergence of powerful digital signal processors for telecommunication and multimedia applications implemented in CMOS VLSI technology creates the need for high-resolution analog-to-digital (A/D) converters that can be integrated in fabrication technologies optimized for digital circuits and systems. However, the same scaling of VLSI technology that makes possible the continuing dramatic improvements in digital signal processor performance also severely constrains the dynamic range available for implementing the interfaces between the digital and analog representation of signals. Oversampled A/D converters based on $\Delta\Sigma$ (delta-sigma) modulation combine sampling at rates well above the Nyquist rate with negative feedback and digital filtering in order to exchange resolution in time for that in amplitude. These converters are especially insensitive to circuit imperfections and component mismatch, and therefore provide a means of exploiting the enhanced density and speed of scaled digital VLSI circuits so as to avoid the difficulty of implementing complex analog circuit functions within a limited analog dynamic range.

A $\Delta\Sigma$ modulator consists of an analog filter and a coarse quantizer enclosed in a feed back loop. Together with the filter, the feedback loop acts to attenuate the quantization noise at low frequencies while emphasizing the high-frequency noise. Since the analog signal is sampled at a frequency much greater than the Nyquist rate and the quantization noise in the low-frequency signal band can be shaped to the high frequency range, a conversion with high resolution may be achieved by removing out-of-band quantization noise with a digital low-pass filter operating on the output of the $\Delta\Sigma$ modulator.

$\Delta\Sigma$ data converters have been successfully applied in many low-frequency fields, such as digital audio and ISDN systems, while efforts are dedicated to developing new $\Delta\Sigma$ structures for higher-frequency applications. A new structure [1] has been proposed for conversion rate up to 2 MHz. The purpose of this thesis is to implement a $\Delta\Sigma$ A/D converter system exploiting that proposed structure in a 0.8-μm CMOS technology, so that it can achieve 12-bit resolution and 2.08 MHz conversion working at 50 MHz clock for 1 MHz signal band.

Chapter 2 discusses the concepts and structures of $\Delta\Sigma$ modulation as well as digital filters for decimation of sampling rate. A general introduction of A/D converters is also given in Chapter 2. Chapter 3 contains the design of the analog $\Delta\Sigma$ modulator. This includes discussion sections on the new structure, VLSI implementation of the structure, and SPICE simulation results. Many analog building blocks, such as folded-cascode opamp, comparator, flash A/D converter, and switched-capacitor integrator, are designed on transistor level. Chapter 4 presents the design of digital filters in Matlab, VHDL modeling and simulation, and synthesis of standard-cell integrated circuits. All of the VHDL codes and Matlab programs are included in the appendices. A conclusion chapter gives some discussions on the future layout design.

Chapter 2

FUNDAMENTAL OF DELTA-SIGMA DATA CONVERTERS

Analog-to-digital interfaces are becoming increasingly more important as translators between state-of-the-art digital signal processing (DSP) systems and the stubbornly analog outside world. With strict demands for higher accuracy in signal processing, these interface stages must also become more precise. In particular, the data converters, both analog-to-digital (A/D) and digital-to-analog (D/A), must be at least as accurate as the overall precision of the DSP systems. At the same time, as both the feature sizes and bias voltages of very-large-scale integrated (VLSI) systems decrease, the accuracy and dynamic range of analog components are reduced, making the fabrication of monolithic high-resolution data converters more difficult.

Compared with the conventional Nyquist-rate converters, oversampling converters, where the sampling and processing rate is much higher than the Nyquist rate, relax the requirements placed on the analog circuitry at the expense of more complicated digital circuitry. This trade-off becomes more desirable for modern VLSI technologies. In order to achieve extra resolution and lower the oversampling ratio (OSR), noise-shaping techniques are usually used in oversampling converters, which is commonly referred to as delta-sigma ($\Delta\Sigma$) modulation [2].[1]

In this chapter, an overview of various A/D converters is presented first. Then the basics of oversampling converters and architectures for $\Delta\Sigma$ modulators are discussed. Finally, digital filters are described for decimating the modulated signal. Since this work is dedicated to a $\Delta\Sigma$ A/D converter (ADC), the issues special for $\Delta\Sigma$

[1] Delta-sigma modulation is also sometimes referred to as sigma-delta modulation.

D/A converters (DAC) will not be covered here. Interested readers may refer to [3, Chapter 12] for details of $\Delta\Sigma$ DAC.

2.1 Overview of Various A/D Converter Types

Analog-to-digital converters can be classified in a number of ways. For example, we may classify them into two categories: Nyquist-rate and oversampling [4, Chapter 10], on the basis of their sampling rate of the input signal. If the bandwidth of the signal is f_0, then the sampling rate f_s has to be at least Nyquist rate $2f_0$ in order to avoid aliasing. In the conventional Nyquist-rate A/D converter, the sampling rate f_s of the analog input signal is the same as the output data rate. Hence, a one-to-one correspondence exists between analog input samples and digital output words. Whereas, in the oversampling A/D converter, the f_s is much higher than Nyquist rate, and the oversampling ratio (OSR) $f_s/(2f_0)$ is usually in the range from 16 to 512. Thus, every output word of the oversampling data converter is found as a weighted average of many consecutive analog input samples.

Since the oversampling A/D converter will be thoroughly discussed in the rest of this chapter, this section mainly discusses some types of Nyquist-rate A/D converters. We may further classify these Nyquist-rate A/D converters into three categories—low-to-medium speed, medium speed, and high speed (Table 2.1) [5, 6]. For comparative purpose, oversampling converter is listed in the table.[2]

It should be noted that the speed and resolution ranges in Table 2.1 just show a rough estimate. The classification criteria may need to be expanded based on the actual design.

A brief look at these Nyquist-rate converters may give us an idea of how they work and where they might fit:

[2] The conversion rate is equivalent to that of a Nyquist-rate converter.

Table 2.1: Different A/D Converter Types.

Classification	Type of A/D Converter	Samples per second	Resolusion Bits
Low-to-Medium Speed, High Accuracy	Integrating (dual-slope)	10–30	12–24
	Oversampling (delta-sigma)	10^3–10^6	12–22
Medium Speed, Medium Accuracy	Sampling (successive approximation)	10^4–10^6	8–16
High Speed, Low-to-Medium Accuracy	Parallel (flash)	10^6–10^9	6–16
	Subranging (two-step, multistep or pipelined)	10^5–10^7	8–16
	Interpolating (non-folding, folding)	10^6–10^8	8–12

Integrating converter:

This is the simplest and a popular approach for realizing high-accuracy data conversion on very slow-moving signals. The basic function of a quantizer is to electronically define a range of input values, subdivide that range into a set of subregions (i.e., 2^n levels for n-bit resolution), and then decide within which subregion the input sample lies. A *dual-slope* integrating converter achieves this by first performing an integration of an unknown input signal ($-V_{in}$) in a fixed time period (T_1), followed by the integration of a known reference signal of opposite polarity (V_{ref}) to examine how long it takes to decay the integration to zero (T_2). The time ratio of T_2/T_1 can be used to determine the V_{in} level. This kind of converters are widely used in measurement instruments such as voltage or current meters because of their simplicity and insensitivity to hardware imperfections. However, the conversion speed is quite slow. For example, the worst-case conversion speed occurs when V_{in} equals V_{ref}. In this case, the conversion is equivalent to going from the lowest level to the highest level then to the lowest level again, which needs 2^{n+1} clock cycles.

Sampling converter:

This is so far the most popular fully integrated solution for moderate performance systems. This works by applying a “binary search” algorithm to successively approximate the input sample with the output of a digital-to-analog converter (DAC), whose input is also the ADC’s output, and test each bit, instead of each level in the integrating converter. A binary search divides the search space in two each time, and the desired input level can be found in n steps for a set of 2^n levels. Specially, in the first clock cycle, the most significant bit (MSB) of the DAC’s input is set to ‘1’ so that the output of the DAC equals half of V_{ref}. The comparison determines whether the MSB should be kept on ’1’ or not. In the second clock cycle, the bit next to the MSB is determined, followed by next bit and so on until the least significant bit (LSB) is determined. Thus, a *successive-approximation* converter requires n clock cycles to complete an n-bit conversion. The cost for this speed-up is that a high-speed DAC with precision on the order of the conversion itself is required.

Parallel converter:

In parallel conversion, all subregions or levels are examined simultaneously using one comparator per subregion. Therefore, it is the highest-speed A/D converter, so called *flash* converter. The result can be obtained in the order of one clock cycle, but the cost in hardware area and power consumption is much greater than successive approximation, especially for higher resolutions (e.g., 8-bit resolution results in $2^8 - 1 = 255$ comparators!). A 3-bit flash ADC will be designed as a part of this thesis work.

Subranging converter:

Very similar to flash converters, except that these perform two or more substeps or multisteps instead of requiring a complete conversion in one cycle. Multistep

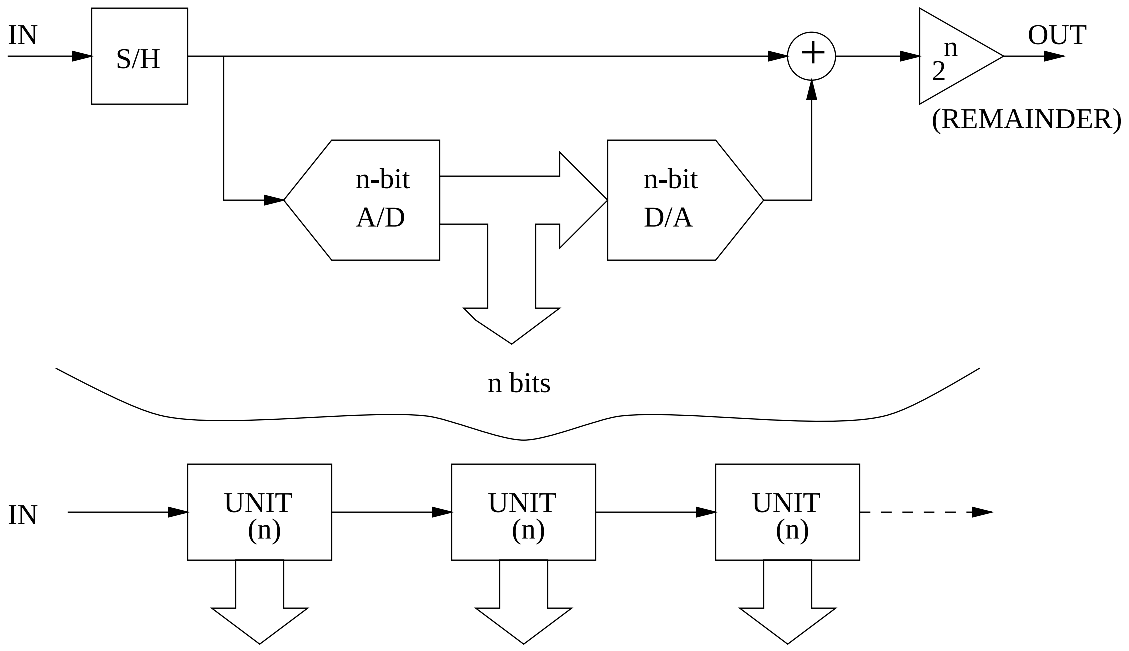


Figure 2.1: Block diagram of pipelined A/D converter.

converter is also referred as to *pipelined* converter. These converters utilize a series of cascaded blocks in which a low-resolution A/D converter is used to estimate the signal, and a D/A converter and an analog subtraction block remove this estimate from the signal. A remainder voltage is passed on to the next stage where the process is repeated. A block diagram is shown in Figure 2.1 [4, Chapter 9]. The low-resolution A/D converter inside each cascade block is usually a flash converter. However, these converters can potentially achieve a throughput rate comparable with parallel converters but at much less hardware cost at large resolutions. It is straightforward to see that subranging converters save a lot of comparators over their flash counterparts. For example, an 8-bit *two-step* converter needs only $2^5 - 2 = 30$ comparators, if each step involves a 4-bit flash converter, a big reduction comparing to 255 comparators required for a parallel converter.

Interpolating converter:

A serious problem that the flash converters face is the large number of opamps attached to the input signal, which introduce quite high input capacitance so that a very big and power-hungry buffer has to be added to drive the signal. Interpolating converters can alleviate this problem by reducing the number of input amplifiers by an interpolating factor of k. For example, in a 4-bit interpolating A/D converter (Figure 2.2) [5, Chapter 13], only 4 input amplifiers may be used instead of 15 in a flash converter (i.e., interpolating factor of 4). These input amplifiers behave as linear amplifiers near their threshold voltages but are allowed to saturate once their differential inputs become moderately large. As a result, noncritical latches (as simple as an inverter) need only determine the sign of the amplifier outputs (i.e., above or below the threshold, which is near the midpoint of the two logic levels). Other subregions are interpolated between adjacent outputs of these amplifiers (e.g., 3 interpolative levels between each two adjacent amplifiers in Figure 2.2). The output of all the latches is encoded by the digital logic circuit. A *folding* architecture [7] is often used to further reduce the large number of latches in the above interpolating converters. A folding A/D converter works like a two-step converter while not requiring an accurate D/A converter.

Other low-speed and low-accuracy A/D converters are not discussed here since the major emphasis in this thesis is on advanced DSP applications in telecommunication. All of the above ADCs except integrating converter have been successfully used in most telecommunication applications. Perhaps the most significant development in telecommunication ADCs in recent years has come from the oversampling delta-sigma approach.

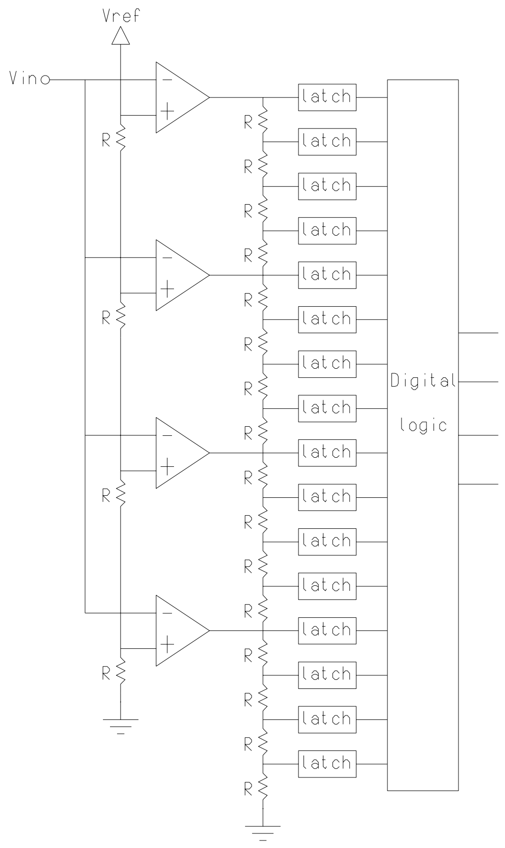


Figure 2.2: A 4-bit interpolating A/D converter.

2.2 Basic Concepts of Delta-Sigma A/D Conversion

As mentioned above, oversampling delta-sigma converters are more tolerable to imprecision of analog circuits by taking advantage of complicated and fast digital circuits. The second advantage of delta-sigma converters is that they simplify the requirements placed on the analog anti-aliasing filters for ADCs and smoothing filters for DACs. Furthermore, a sample-and-hold is usually not required at the input of the oversampling ADC. Therefore, they have become popular in recent years at least for medium to low speed applications such as high fidelity digital audio [8, 9, 10], digital telephony [11, 12, 13], and instrumentation. Future applications in digital video and digital radar systems are imminent as faster technologies become available.

2.2.1 A/D Conversion Terms

Before detailed concepts of $\Delta\Sigma$ modulation are discussed, some commonly used terms describing the performance of data converters need to be reviewed.

Resolution The number of bits of resolution refers to the smallest analog input level to cause change in the digital output word. Thus, an n-bit resolution implies that the converter can resolve 2^n distinct analog levels.

LSB (Least Significant Bit) One LSB=(full-scale input voltage)/resolution. For example, with input full scale= $5V$ and resolution=10 bits, then 1 LSB= $5/2^{10} = 4.9mV$.

Offset Error For an A/D converter, the offset error is defined as the relative deviation of the actual $V_{0\ldots01}$ from $\frac{1}{2}$ LSB, or mathematically,

$$E_{off} = \frac{V_{0\ldots01}}{V_{LSB}} - \frac{1}{2}$$

in units of LSBs, where $V_{0\ldots01}$ refers to the input voltage as the output data becomes $0\ldots01$.

Gain Error This is also called the *full-scale* error, which refers to the difference between the actual input that produces a full-scale output word and the ideal voltage when the offset error has been reduced to zero. Or mathematically,

$$E_{gain} = \frac{V_{1\ldots11} - V_{0\ldots01}}{V_{LSB}} - (2^n - 2)$$

in units of LSBs, where $V_{1\ldots11}$ refers to the input voltage as the output data becomes $1\ldots11$.

Differential Nonlinearity (DNL) Error In an ideal converter, each analog step size is equal to 1 LSB. After the offset and gain errors have been removed, DNL error is defined as the variation in analog step size away from 1 LSB. Thus, a converter with a maximum DNL of 0.5 LSB has its step size varying from 0.5 LSB to 1.5 LSB. For an ADC, when the DNL reaches 1 LSB or greater, there will be missing output codes.

Integral Nonlinearity (INL) Error The INL is the worst-case deviation from a straight line between zero and full-scale endpoints of the converter's transfer response, expressed in LSBs. An ADC is guaranteed not to have any missing codes if the maximum INL error is less than 0.5 LSB.

Accuracy The *absolute accuracy* of a converter is defined to be the difference between the expected and actual transfer responses, which includes the offset, gain, and nonlinearity errors. The *relative accuracy* is the accuracy after the offset and gain errors have been removed, which is usually referred to as the *maximum INL error*. Accuracy can be expressed as a percentage error of full-scale value, as the effective number of bits, or as a fraction of an LSB. For example, in order to achieve $\frac{1}{2}$ LSB integral linearity, an 8-bit accuracy implies $1/2^8 \approx 0.4\%$ matching, while a 12-bit accuracy implies $1/2^{12} \approx 0.025\%$ matching. *Note that a converter may have 12-bit resolution with only 10-bit accuracy, or 10-bit resolution with 12-bit accuracy. An accuracy greater than the resolution means that the converter's transfer response is very precisely controlled.*

Signal-to-Noise Ratio (SNR) This is the ratio of the original input signal power to the background noise power, or equivalently, the rms ratio of the input signal amplitude to the noise amplitude. For an ideal ADC with a sinusoidal input, the SNR related to the resolution n is

$$SNR_{rms} = 6.02\,n + 1.76 \quad dB \tag{2.1}$$

Dynamic Range The *dynamic range* of a converter is usually specified as the ratio of the rms value of the *maximum* amplitude of the sinusoidal input signal to the rms output noise plus the distortion measured when the same sinusoid is present at the output. The rms output noise plus distortion is obtained by first eliminating the sinusoid from the measured output. Dynamic range can also be expressed as an effective number of bits using the relationship presented in Equation (2.1). Essentially, this is an indication of how far it is possible to go below the full-scale input signal without hitting noise and/or distortion.

2.2.2 Oversampling Without Noise Shaping

The notion of using artificially high sampling rates and 1-bit quantization to achieve high resolution A/D conversion at a lower rate has been of interest ever since *delta modulation* [14] was first proposed in 1946. Extra dynamic range and SNR can be obtained by spreading the quantization noise power over a larger frequency range.

Quantization Noise

Quantization of amplitude and sampling in time are at the heart of all digital modulators. Periodic sampling at rates more than twice the signal bandwidth need not introduce distortion, but quantization does. A quantizer can be modeled as adding *quantization error* $e(n)$ to input $x(n)$ to generate output $y(n)$, i.e., $y(n) = x(n)+e(n)$, where n refers to n-th sample. The quantization error is the difference between the input and output values, which is bounded by $\pm\Delta/2$, where Δ equals the differences between two adjacent quantization levels, i.e., 1 LSB.

The error $e(n)$ is completely defined by the input $x(n)$, but if $x(n)$ is very active, $e(n)$ can be approximated as an independent random number uniformly distributed between $\pm\Delta/2$. Thus we can treat the quantization error as white noise e with power:

$$P_e = \frac{1}{\Delta}\int_{-\Delta/2}^{+\Delta/2} e^2\,de = \frac{\Delta^2}{12} \tag{2.2}$$

and it is independent of the sampling frequency, f_s. Also the spectral density of e is white (i.e., a constant over frequency) and all of its power folds into the frequency band $\pm f_s/2$ (a two-sided representation of frequencies). Then the spectral density of the sampled noise is given by

$$E(f) = \sqrt{\frac{P_e}{f_s}} = (\frac{\Delta}{\sqrt{12}})\sqrt{\frac{1}{f_s}} \tag{2.3}$$

Oversampling Advantage

Oversampling occurs when the signals of interest are bandlimited to f_0 yet the sampling rate is at f_s, where f_s is much larger than $2f_0$. As it was defined in the sections above, the oversampling ratio, $OSR \equiv \frac{f_s}{2f_0}$.

After quantization, since the signals of interest are all below f_0, the quantized signal (along with noise) is filtered by

$$H(f) = \begin{cases} 1 & |f| \leq f_0 \\ 0 & f_0 < |f| < f_s \end{cases} \tag{2.4}$$

This filter eliminates quantization noise (together with any other image signals) greater than f_0.

While the in-band signal power is the same as the original input signal power, the quantization noise power is reduced to

$$P_e^{'} = \int_{-f_s/2}^{f_s/2} E^2(f)|H(f)|^2\, df = \int_{-f_0}^{f_0} E^2(f)\, df = \frac{2f_0}{f_s}P_e = \frac{P_e}{OSR} \tag{2.5}$$

Therefore, each doubling of OSR (i.e., sampling at twice the rate) decreases the in-band noise power by one-half or, equivalently, 3 dB, increasing the resolution by only 0.5 bits according to Equation (2.1).

2.2.3 Noise-Shaping Oversampling $\Delta\Sigma$ Modulation

A more efficient oversampling quantizer is the noise-shaped $\Delta\Sigma$ modulator shown in Figure 2.3(a). Although most present $\Delta\Sigma$ converters make use of 1-bit

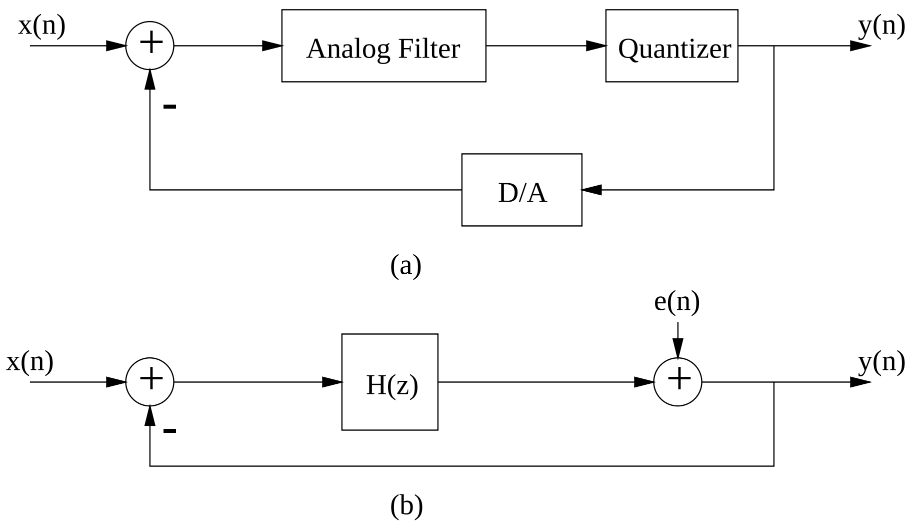


Figure 2.3: A $\Delta\Sigma$ modulator and its linear model.

quantizers (i.e., only two output levels) due to the inherent linearity between two levels, a general discussion may be addressed on multilevel quantizers. By analyzing the linear model shown in Figure 2.3(b) as having two *independent* inputs $x(n)$ and $e(n)$, which is an approximation for active input signal $x(n)$, we can derive a signal transfer function, $S_{TF}(z)$, and a noise transfer function, $N_{TF}(z)$, as

$$S_{TF}(z) \equiv \frac{Y(z)}{X(z)} = \frac{H(z)}{1+H(z)} \tag{2.6}$$

$$N_{TF}(z) \equiv \frac{Y(z)}{E(z)} = \frac{1}{1+H(z)} \tag{2.7}$$

Note that the zeros of $N_{TF}(z)$ will be equal to the poles of $H(z)$. In other words, when $H(z)$ goes to infinity, $N_{TF}(z)$ will go to zero.

In the frequency domain, the output signal can be written as the combination of the input signal and the noise signal:

$$Y(z) = S_{TF}(z)X(z) + N_{TF}(z)E(z) \tag{2.8}$$

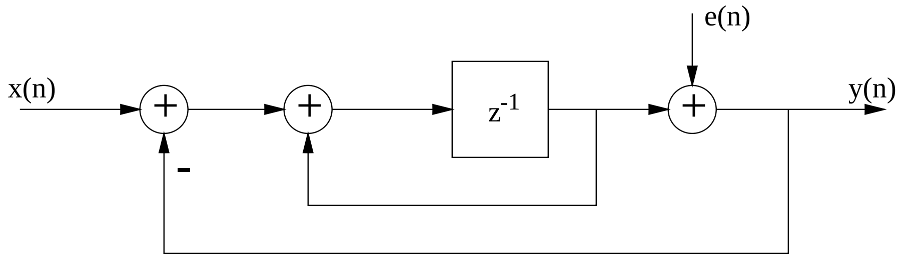


Figure 2.4: A first-order ΔΣ modulator.

To noise-shape the quantization noise in a useful manner, the magnitude of $H(z)$ should be so large over the frequency band of interest as to make $N_{TF}(z)$ near zero and $S_{TF}(z)$ near unity. Thus, the quantization noise is reduced over the signal bandwidth while the signal itself is largely unaffected. The feedback will not reduce the high-frequency noise since there is little loop gain at high frequency. However, the out-of-band noise can be removed by additional low-pass digital filters.

First-Order Noise Shaping

To realize first-order noise shaping, $N_{TF}(z)$ should have a zero at dc (i.e., $z = e^{j\omega T} = 1$, where $\omega = 0$) so that the quantization noise is high-pass filtered. Since the zeros of $N_{TF}(z)$ are equal to the poles of $H(z)$, $H(z)$ should also have a pole at $z = 1$, resulting in a choice of a discrete-time integrator:

$$H(z) = \frac{1}{z-1} \tag{2.9}$$

A block diagram for such a choice is shown in Figure 2.4. The signal goes first through a subtractor and then an accumulator, corresponding to ΔΣ .

From a time domain point of view, the feedback forces the average value of the quantized output $y(n)$ to equal the average value of input $x(n)$ so that the integrator's input $x(n)-y(n)$ equals zero (otherwise, the infinite dc gain will amplify the difference to infinity). This configuration has similarity with an ideal opamp having unity close-loop gain.

From Equations (2.6), (2.7), and (2.9), we have $S_{TF}(z) = z^{-1}$ and $N_{TF}(z) = (1 - z^{-1})$, corresponding to a one-clock-period delay for signal and a first-order high-pass filter for noise. The spectral density of the modulation noise may be expressed as

$$E^{'}(f) = E(f)|N_{TF}(z)| = E(f)|1 - z^{-1}| = E(f)|1 - \epsilon^{-j\omega T}| = \sqrt{8P_eT}\sin\frac{\omega T}{2} \quad (2.10)$$

where $\omega = 2\pi f$, $\frac{1}{T} = f_s$, and the relation of $E(f)$ and P_e is defined in Equation (2.3). Hence, the in-band noise power can be calculated as

$$P_e^{'} = \int_{-f_0}^{f_0} E^{'2}(f)\,df \cong P_e\frac{\pi^2}{3}(\frac{1}{OSR})^3 \quad (2.11)$$

for $f_0 \ll f_s$ (i.e., $OSR \ll 1$). The final SNR for a sinusoidal input signal can be given [3, Chapter 2] as:

$$SNR_{rms} = 6.02n + 1.76 - 5.17 + 30\log(OSR) \quad (2.12)$$

We see here that doubling the OSR reduces the noise by 9 dB or provides 1.5 bits of extra effective resolution for the first-order $\Delta\Sigma$ modulator.

Second-Order Noise Shaping

A major shortcoming of the simple $\Delta\Sigma$ system shown in Figure 2.4 is that it may generate low-frequency (and thus in-band) tones for special values of the input $x(n)$. To see how this may occur, assume for the moment that we are using 1-bit quantizer and 1-bit DAC in the system, and the two output levels of the 1-bit DAC are $0V$ and $1V$, while the input $x(n)$ is a constant voltage k/m volt where k and m are relatively prime integers and $k < m$. Then, it is possible to have a periodic DAC output pattern with each period m pulses long and containing k pulses of value $1V$

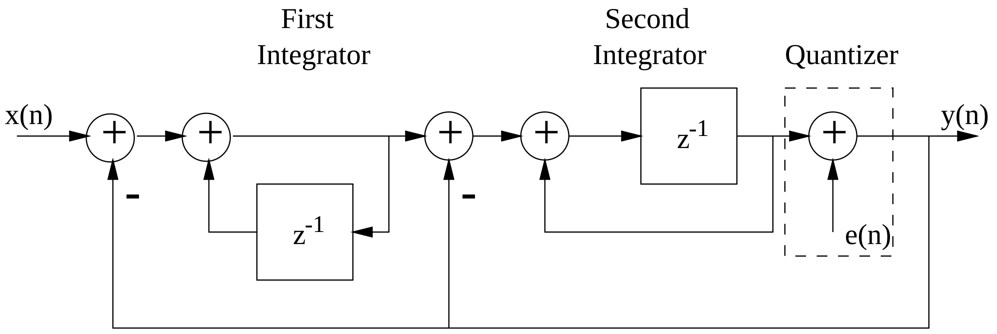


Figure 2.5: A second-order $\Delta\Sigma$ modulator.

and $(m-k)$ pulses of $0V$. The average value of the DAC output will then equal $x(n)$, and hence the loop can settle into a steady state oscillation under these conditions. If m is sufficiently large so that $m > f_s/f_0 = 2OSR$, then the fundamental component f_s/m of the oscillation wave will fall into the signal band, and it may therefore appear as a sinusoid tone of considerable amplitude in the final data output, which is often called *pattern noise* [15].

One way to avoid the pattern noise is to inject an extra noise (call *dither*) into the loop, which will also reduce the dynamic range of the system [16]. Another way is to reduce the correlation between $x(n)$ and $e(n)$ by using a higher-order loop filter, which will also result in more selective noise shaping, and thus in an improved SNR. The modulator shown in Figure 2.5 realizes second-order noise shaping by using two cascaded integrators. For this modulator, the signal transfer function can be derived as $S_{TF}(z) = z^{-1}$, corresponding to a clock cycle delay, and the noise transfer function is $N_{TF}(z) = (1 - z^{-1})^2$, a second-order high-pass filter.

Similar calculations as in the Section 2.2.3 may give the noise power and SNR for sinusoidal signal as:

$$P_e^{'} \cong P_e \frac{\pi^4}{5} (\frac{1}{OSR})^5 \tag{2.13}$$

$$SNR_{rms} = 6.02n + 1.76 - 12.9 + 50 \log{(OSR)} \tag{2.14}$$

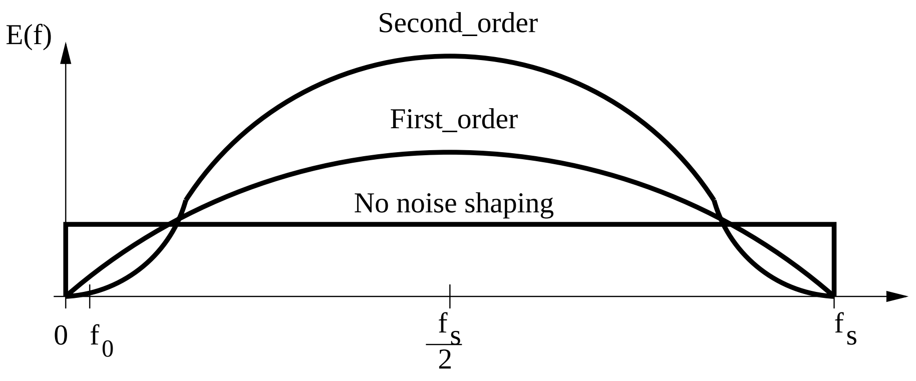


Figure 2.6: Spectral density of noise for some different noise-shaping modulation.

Therefore, doubling the OSR improves the SNR for a second-order modulator by 15 dB or, equivalently, an extra resolution of 2.5 bits/octave. A comparison for the spectral densities of noise among zero-, first-, and second-order noise-shaping modulators is shown in Figure 2.6. The noise power decreases over the signal band (i.e., from 0 to f_0) as the noise-shaping order increases. However, the out-of-band noise increase for the higher-order modulators.

Furthermore, the generation of in-band tones is largely prevented by the added feedback path which reduces the dependence of the quantization noise on the recent values of input signal, and thus decrease their correlation. In fact, the feedback signal added to the input of the second integrator acts as a dither signal.

2.3 Architectures of $\Delta\Sigma$ A/D Converters

The world of $\Delta\Sigma$ converters can be roughly divided into the following camps: single-bit single-stage low-order designs, single-bit single-stage high-order designs, multi-stage cascaded designs with feedforward error cancellation, and multibit noise shapers.

The systems shown in Figure 2.4 and 2.5 are in the first categary as long as 1-bit quantizers are used. They have guaranteed stability [3, Chapter 3] with little restriction on input range and simple circuit design. However, they cannot achieve high SNR with low-to-medium oversampling ratios.

2.3.1 Higher-Order Single-Stage $\Delta\Sigma$ Converters

In principle, arbitrary higher-order loop filters can be used and configured in cascade structure like that in Figure 2.5. When a modulator has L loops and is not overloaded, linear analysis [16] can show that the spectral density and in-band power of the modulation noise are

$$E^{'}(f) = \sqrt{2P_eT}[2\sin(\frac{\omega T}{2})]^L \tag{2.15}$$

$$P_e^{'} \cong P_e \frac{\pi^{2L}}{2L+1}(\frac{1}{OSR})^{2L+1} \tag{2.16}$$

This noise falls $3(2L-1)$ dB for every doubling of OSR, providing $(L-\frac{1}{2})$ extra bits of resolution.

However, the stability of the loop becomes precarious for loop filters of order $L > 2$. Linearized analysis, which results in $N_{TF}(z) = (1 - z^{-1})^L$, is not a reliable predictor of the stability, since the 1-bit quantizer is a grossly nonlinear element whose equivalent gain varies abruptly with the value of its input. It is revealed [3, Chapter 4] that for guaranteed stability the equivalent quantizer gain must be high. Thus, with the fixed output amplitude, this will be achieved only if the quantizer input is small. To ensure this, the maximum amplitude of input must be restricted to pretty low, so as to inversely decrease the dynamic range.

In practice, the loop filter has to be carefully designed and the stability may be signal dependent. The coefficients of the filter must come from a discrete-time simulation rather than from a linear model. Another challenging aspect of stability analysis lies in establishing a "safe" input range, which is also related to the choice of the filter.

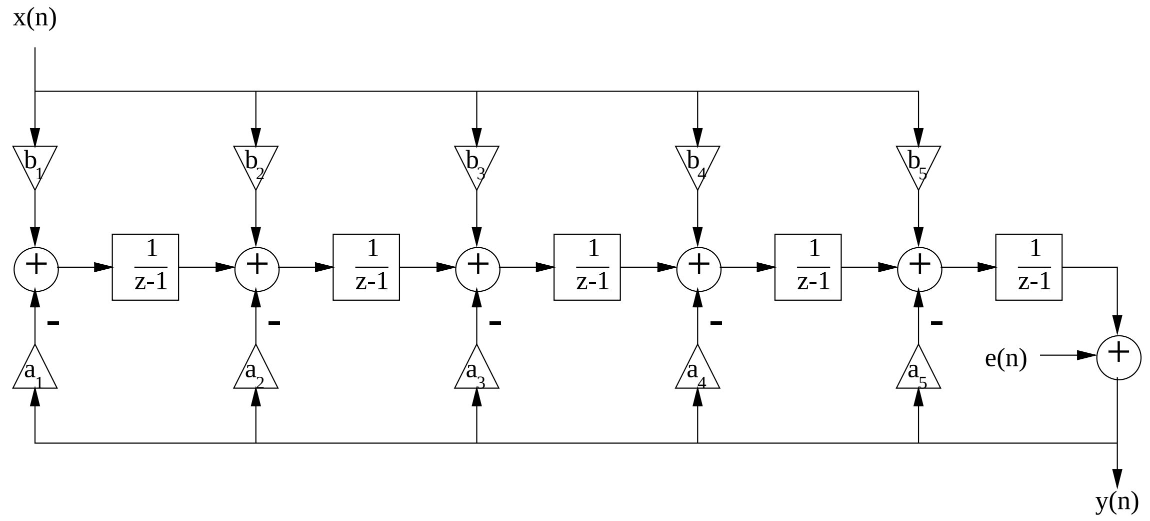


Figure 2.7: Chain of integrators with distributed feedback and distributed feedforward inputs.

There are many loop topologies to build a high-pass filter for modulation noise. One [17] is using chain of integrators with distributed feedback and distributed feedforward inputs as shown in Figure 2.7. This fifth-order modulator has the noise transfer function and signal transfer function as:

$$N_{TF}(z) = \frac{1}{1 - \frac{a_5}{z-1} - \frac{a_4}{(z-1)^2} - \ldots - \frac{a_1}{(z-1)^5}} = \frac{(z-1)^5}{D(z)} \tag{2.17}$$

$$S_{TF}(z) = \frac{b_1 + b_2(z-1) + \ldots + b_5(z-1)^4}{D(z)} \tag{2.18}$$

where $D(z) = (z-1)^5 - a_5(z-1)^4 - \ldots - a_1$. The basic idea of adding $D(z)$ is to introduce poles to flatten the high-frequency portion of $N_{TF}(z)$, so that the quantizer gain will not be too low to result in stability at high frequency.

2.3.2 Multi-stage (Cascade) $\Delta\Sigma$ Converters

An alternative structure for realizing higher-order noise-shaping converters, which is free of the stability problems associated with the higher-order single-stage

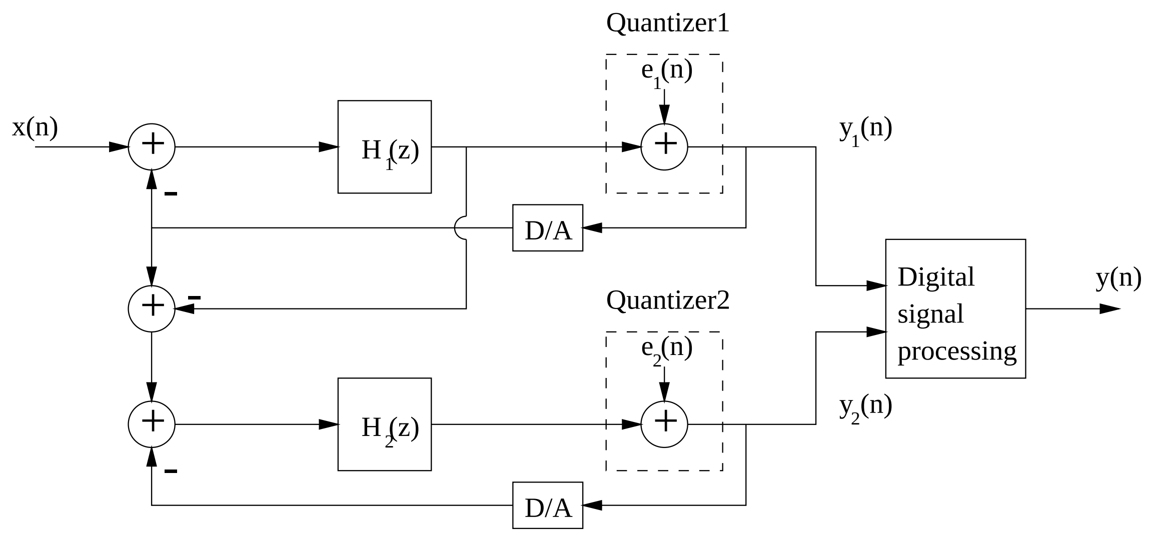


Figure 2.8: Block diagram of cascaded modulator.

converters described above, is the multi-stage or cascade architecture as shown in Figure 2.8. The overall $\Delta\Sigma$ modulator consists of a cascade of several lower order single-loop modulators, each with its own quantizer. Each single-loop modulator in the cascade converts the quantization error from the preceding modulator. The error of all but the last single-loop modulator are then digitally canceled. The guaranteed stability is achieved by using first- and/or second-order loops in a feedforward (as opposed to feedback) configuration.

Since the design in this thesis is also a cascaded modulator, a detailed analysis will be given in the next chapter. It should be mentioned here that a major disadvantage of the cascaded structure is that the exact cancellation of the error $e_1(n)$ requires accurate matching of the analog transfer functions $H_1(z)$ and $H_2(z)$ to some digital functions determined by the DSP system. If these conditions are not exactly satisfied, then unfiltered or poorly filtered noise due to $e_1(n)$ will leak into the output data $y(n)$, and the SNR plummets.

2.3.3 Multi-bit $\Delta\Sigma$ Converters

As mentioned before, one-bit $\Delta\Sigma$ modulators employ a 1-bit internal DAC with inherent linearity that does not require precision component matching. This relax of requirement on analog components is a great attractiveness for modern VLSI technologies. However, it is easy to see from Equation (2.12) and (2.14) that employing multibit quantizers in the modulators shown in Figure 2.4 and 2.5 can dramatically increase SNR by 6 dB per additional bit. This is because the Δ in Equation (2.2), which is the level spacing of the quantizer, will decrease by a factor of 2 per bit so that the noise power P_e decreases by a factor of 4 per bit. Equivalently, the multibit $\Delta\Sigma$ coder can achieve resolution comparable to that of a single-bit modulator at a lower sample rate, which is a significant advantage in applications requiring high bandwidth like digital video. The lower clock rate possible within the multibit modulators may decrease power consumption in the digital circuitry. Finally, the multibit quantizer is a better approximation to a linear amplifier than a single-bit one; hence, the stability properties are better and the agreement between the behavior predicted by linear theory and the actual performance is improved. Therefore, multibit $\Delta\Sigma$ converters are gaining popularity [18, 1, 19].

The multibit internal A/D converter must be a parallel (flash) type circuit, since stability and noise cancellation allow only one clock period for conversion. On the other hand, the ADC nonlinearity merely increases the quantization noise somewhat, and will be suppressed by the noise shaping process. By contrast, any nonlinearity of the internal feedback DAC will directly affect the output signal. This can be seen from the analysis of the linear model shown in Figure 2.9, where $a(n)$ represents the errors caused by the deviation of the quantizer thresholds from their ideal values, i.e., ADC nonlinearity, and $d(n)$ represents the errors due to the internal DAC nonlinearity. For noise shaping to occur, the gain of $H(z)$ must be large at low frequencies as we saw in Section 2.2.3. Therefore, both $a(n)$ and $e(n)$ are reduced by

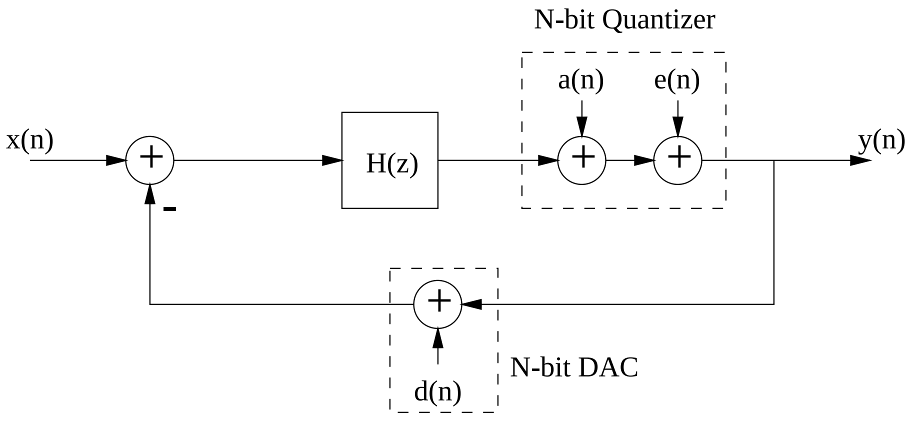


Figure 2.9: Simplified linear model of a $\Delta\Sigma$ ADC with nonlinearity errors.

this large gain when referred back to the input $x(n)$. However, $d(n)$ still resides in the feedback path so that the ultimate linearity of $y(n)$ is no better than the linearity of the N-bit internal DAC. For example, 16-bit resolution of a multibit $\Delta\Sigma$ converter can only be achieved if the internal DAC has an accuracy of $1/2^{16} \cong 0.0015\%$. A direct realization of such high-precision converter (e.g. laser-trimmed resistors) is very expensive and defeats the virtue of using oversampling.

There are several alternative techniques [3, Chapter 8] for reducing the effects of DAC nonlinearity. However, a novel structure using two quantizers (Figure 2.10, proposed by Leslie and Singh [18], is easier to implement. One of the two quantizers is a single-bit circuit contained in a $\Delta\Sigma$ loop, which includes a single-bit DAC in the feedback path. Since this DAC plays the key role in determining the linearity of the modulator, its inherent linearity is used to full advantage. An added path with a second quantizer, a multibit one, is used to convert and cancel the large quantization error generated by the single-bit quantizer. This scheme is conceptually similar to the cascade technique. Analysis of the system of Figure 2.10(b) gives for the output signal the expression as:

$$Y = H_1V_1 + H_2V_2 = S_{TF}(H_1 + H_2)X + (H_1N_{TF} + H_2N_{TF} - H_2)E_1 + H_2E_2$$

Here, S_{TF} and N_{TF} are the signal and noise transfer functions, respectively, of the 1-bit loop; E_1 and E_2 are the quantization errors of the 1-bit and N-bit ADCs, respectively, in the z-domain. An appropriate choice of the $H(z)$, $H_1(z)$, and $H_2(z)$ can cancel the large error E_1 by satisfying the condition

$$\frac{H_1}{H_2} = \frac{1}{N_{TF}} - 1 \tag{2.19}$$

It is usually advantageous to choose S_{TF} and the overall signal transfer function $S_{TF}(H_1 + H_2)$ both as delays of k clock period, i.e., z^{-k}. Then the design equations become $H_1 = 1 - N_{TF}$, $H_2 = N_{TF}$, and

$$Y = z^{-k}X + N_{TF}E_2$$

In practice, condition (2.19) cannot be exactly satisfied due to the inaccuracy of analog N_{TF}. This inaccuracy generally makes it useless to choose the value of N larger than 3-5 bits.

2.4 Digital Filters for $\Delta\Sigma$ A/D Converters

As seen in previous sections, the techniques for $\Delta\Sigma$ A/D converters are heavily based on principles of the digital signal processing filters, which follow with the analog modulator to remove the shaped out-of-band noise, reduce the sampling rate to Nyquist rate, and increase 1-bit or several-bit data word to high-resolution sample word. The block diagram of a whole $\Delta\Sigma$ A/D converter system is shown in Figure 2.11. In this system, $x(n)$ and $y(n)$ are working on oversampling rate, while $y(m)$ is working on Nyquist rate. The process of converting a signal from a given rate to a different rate is called *sampling-rate conversion.* In a $\Delta\Sigma$ A/D converter, the process of reducing the sampling rate is called *decimation*, while in a $\Delta\Sigma$ D/A converter, the process of increasing the sampling rate is called *interpolation.*

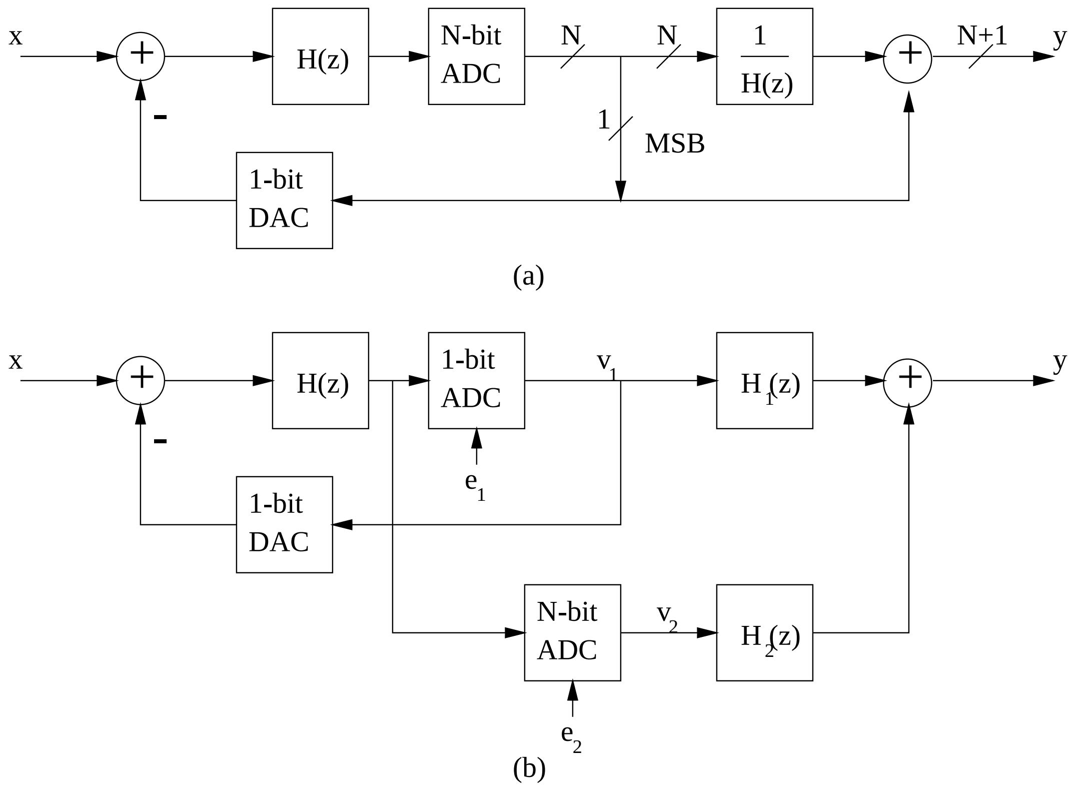


Figure 2.10: (a) Leslie-Singh structure. (b) An Equivalent representation.

2.4.1 Basic Principles of Decimation

The process of decimation in the digital domain can be viewed as a linear filtering operation as shown in Figure 2.12. The input signal $x(n)$ is characterized by the sampling rate f_s, and the output signal $y(m)$ is characterized by f_s/M, where M is an integer factor. The spectrum of $x(n)$, $X(\omega)$, is assumed to be nonzero in the frequency interval $0 \leq |\omega| \leq \pi$ or, equivalently, $|f| \leq f_s/2$. To avoid aliasing [20], the bandwidth of $x(n)$ must first be reduced to $f_{max} = f_s/2M$ or, equivalently, $\omega_{max} = \pi/M$, then the band-limited signal $w(n)$ can be down-sampled by simply discarding $M-1$ out of every M samples to produce the output $y(m)$.

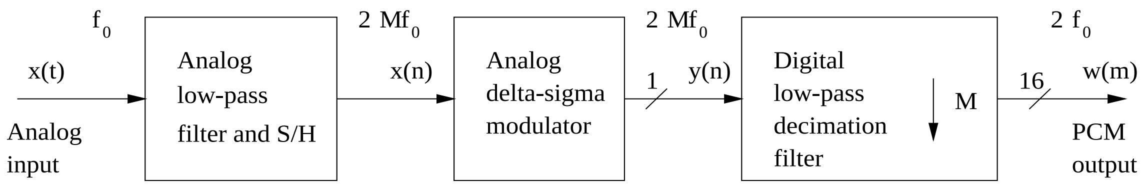

f0
x(t)
Analog input
Analog low-pass filter and S/H
2 Mf0
x(n)
Analog delta-sigma modulator
2 Mf0
1
y(n)
Digital low-pass decimation filter
M
2 f0
16
w(m)
PCM output

Figure 2.11: A $\Delta\Sigma$ A/D converter system.

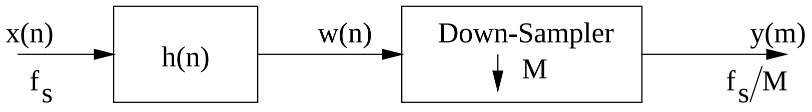


Figure 2.12: Decimation by a factor M.

The filter is a low-pass filter characterized by the impulse response $h(n)$ and a frequency response $H_M(\omega)$, which ideally satisfies the condition

$$H_M(\omega) = \begin{cases} 1 & |\omega| \leq \frac{\pi}{M} \\ 0 & otherwise \end{cases}$$

Thus the filter eliminates the spectrum of $X(\omega)$ in the range $\pi/M < \omega < \pi$. Of course, the implication is that only the frequency components of $x(n)$ in the range $|\omega| \leq \pi/M$ are of interest in further processing of the signal.

In practice, the digital filter, while specified at the high sampling rate f_s, is actually implemented at the low rate f_s/M. This is shown by the relationship

$$y(m) = w(mM) = \sum_{k=0}^{\infty} h(k)x(mM - k)$$

Only one out of every M samples of $x(n)$ needs to be convoluted with $h(n)$.

2.4.2 Multistage Implementation

An important criterion in the design of an A/D or D/A converter is the efficiency in which the decimator or interpolator operation can be implemented. This efficiency is directly related to the type, the order, and the architecture of the digital filter used in the implementation. In practical applications, the decimation factor M is much larger than unity (i.e., $OSR \gg 1$), so a single-stage implementation (i.e., only one filter to do the decimation) will require a very high-order filter and may be computationally inefficient. However, a multistage implementation, which is a cascade of J decimators with $M = \prod_{i=1}^{J} M_i$ and the sampling rate at the output of the ith stage $F_i = F_{i-1}/M_i$, for $i = 1, 2, \ldots, J$, can save pretty much computation and thus is used in most $\Delta\Sigma$ converters.

In a multistage implementation, to ensure that no aliasing occurs in the overall decimation process, each filter stage should be designed to avoid aliasing within the frequency band of interest (e.g., from 0 to $2f_0$). Let us define the passband cutoff frequency $F_{pc} = f_0$ and the stopband cutoff frequency $F_{sc} = F_s/2M$ for the overall decimator.[3] Then aliasing in the band $0 \le F \le F_{sc}$ is avoided by selecting the frequency bands of each filter stage as

$$\begin{array}{ll} passband: & 0 \le F \le F_{pc} \\ transitionband: & F_{pc} \le F \le F_i - F_{sc} \\ stopband: & F_i - F_{sc} \le F \le \frac{F_{i-1}}{2} \end{array} \tag{2.20}$$

The length or order of a FIR filter may be estimated from one of the well-known formulas given in the literature [20]:

$$N = \frac{D_\infty(\delta_p, \delta_s) - f(\delta_p, \delta_s)(\Delta f)^2}{\Delta f} + 1 \tag{2.21}$$

$$\begin{aligned} D_\infty(\delta_p, \delta_s) &= [0.005309(\log \delta_p)^2 + 0.07114(\log \delta_p) - 0.4761] \log \delta_s \\ &\quad -[0.00266(\log \delta_p)^2 + 0.5941(\log \delta_p) + 0.4278] \\ f(\delta_p, \delta_s) &= 11.012 + 0.51244(\log \delta_p - \log \delta_s) \\ \Delta f &= \frac{F_{sc} - F_{pc}}{F_s} \end{aligned}$$

[3] In a practical filter design, a transition band has to be allowed, i.e., $F_{sc} > F_{pc}$.

where δ_p and δ_s represent the passband ripple and stopband ripple, respectively. It is easy to see that in a single-stage implementation, the transition bandwidth becomes small relative to F_s (e.g., usually $\Delta f \ll 1$), leading to excessively large filter orders and high-word-length requirements on the decimator. Whereas, in a multistage implementation, each stage's Δf (i.e., $(F_i - F_{sc} - F_{pc})/F_{i-1}$) is not so small that a relative low-order filter can be used. The total orders of all cascaded filters may be much smaller than the orders of the single filter. An example will illustrate this point in Section 4.1.

Although, in principle, an arbitrary number of stages may be used, practical considerations sometimes lead to the conclusion that a two-stage design is best. A further careful consideration of the first stage filter design can relax the filter requirement to a multiple stopband filter design [3, Chapter 13], such as a *comb* or $sinc^K$ filter, which will save more computations than a conventional FIR design.

2.4.3 Filter Structures

In principle, any number of classical filter design techniques can be applied to make the decimator or interpolator. However, because of the multirate and/or multistage considerations, many of these classical techniques are often ruled out in favor of designs that can take better advantage of the above multirate criteria and achieve a more effective design. These considerations often lead to the use of architectures that minimize the coefficient word lengths, eliminate the need for dedicated high-speed parallel multiplier, or reduce the memory storage requirements.

FIR and IIR Filters

FIR (Finite Impulse Response) and IIR (Infinite Impulse Response) filters are two basic types of digital filters. In many applications, such as high-quality digital audio, linear phase is important; hence, FIR filtering is used exclusively. In voiceband

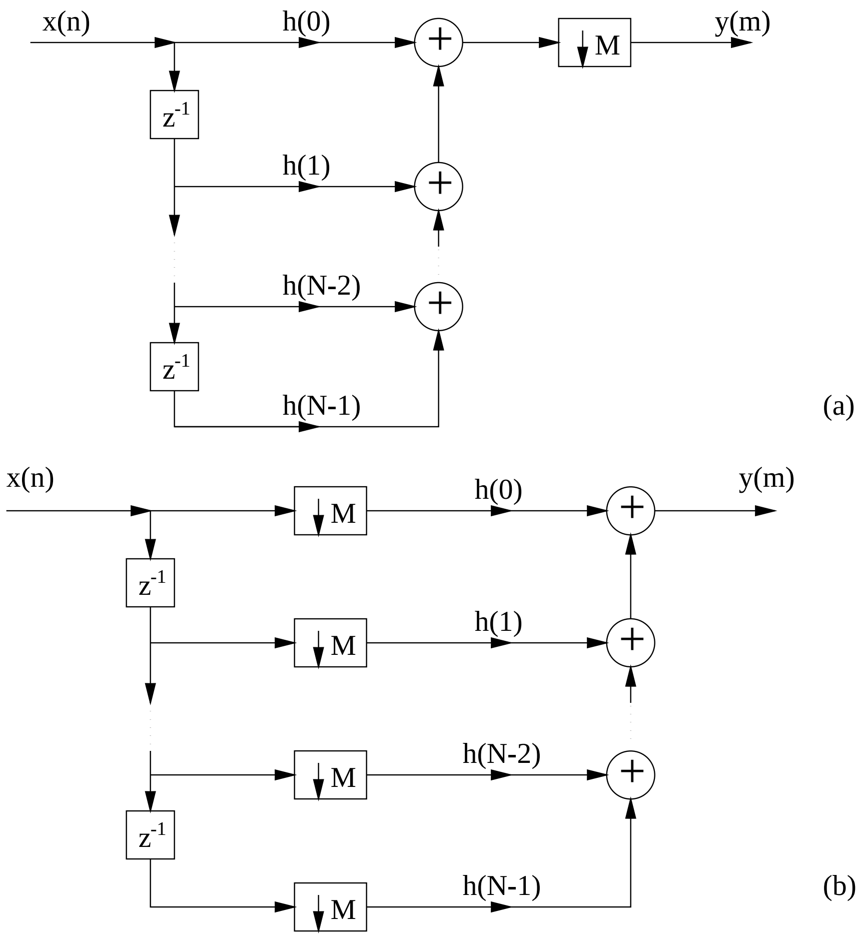


Figure 2.13: FIR structures for decimation by a factor M. (a) Direct form. (b) Efficient direct form.

telephony applications, however, linear phase is not required so that IIR filters may be used as intermediate-stage decimators because they typically require lower order than FIR at the expense of higher internal word lengths for coefficients. In my work, FIR filters are designed due to the assumed linear phase requirement.

In principle, the simplest realization of a decimator is the direct-form FIR structure as shown in Figure 2.13(a) with system function

$$H(z) = \sum_{k=0}^{N-1} h(k) z^{-k} \tag{2.22}$$

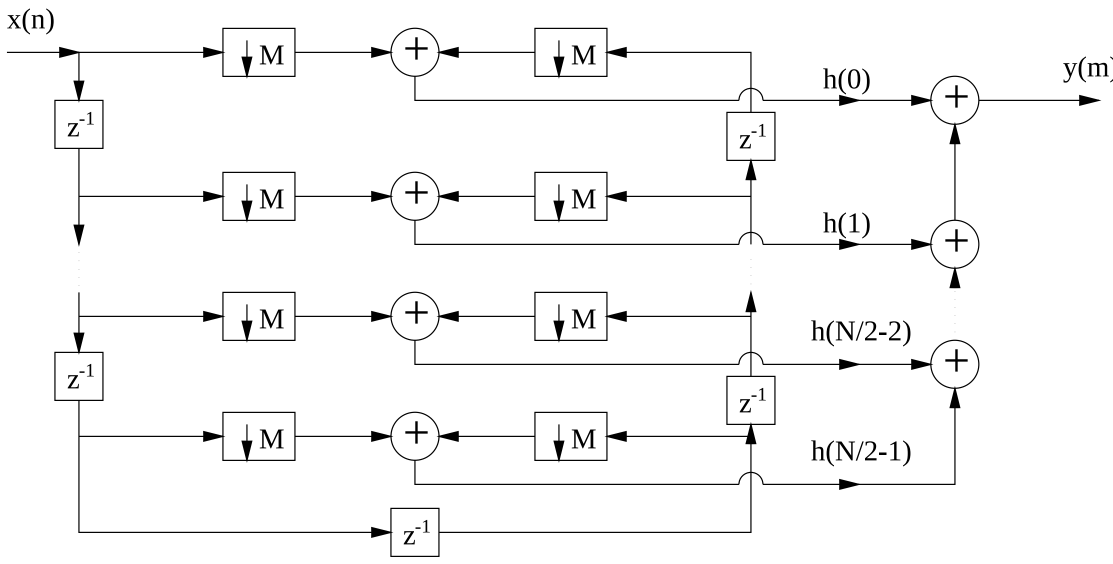


Figure 2.14: Symmetrical FIR structure for decimation by a factor M.

where $h(k)$ is the unit sample response of the FIR filter. Any of the standard, well-known FIR filter design techniques (e.g., window methods, Parks-McClellan algorithm) [21] may be used to carry out the coefficients. Although the direct-form FIR filter realization is simple, it is also very inefficient. The inefficiency results from the fact that the down-sampling process requires only one out of every M output samples at the output of the filter. Consequently, only one out of every M possible values at the output should be computed. Thus, the logical solution to this problem is to embed the down-sampling operation within the filter as illustrated in Figure 2.13(b). Additional reduction in computation can be achieved by exploiting the symmetry characteristics of $h(k)$ as shown in Figure 2.14.

$sinc^K$ Filters

The transfer function for a $sinc^k$ decimation filter has the general form

$$H(z) = (\frac{1}{M}\sum_{i=0}^{M-1} z^{-i})^K = (\frac{1}{M}\frac{1-z^{-M}}{1-z^{-1}})^K \qquad (2.23)$$

and its frequency response is therefore

$$|H(e^{j\omega})| = (\frac{1}{M}\frac{\sin(\omega M/2)}{\sin(\omega/2)})^K = (\frac{sinc(\omega M/2)}{sinc(\omega/2)})^K \tag{2.24}$$

where $\omega = 2\pi f/f_s$. It has $M/2$ spectral zeros if M is even, or $\lceil M/2 \rceil - 1$ zeros if M is odd, at frequencies that are multiples of the decimated sampling frequency ω_d. These zeros correspond to multiple notches or stopbands which are wide enough to cover the required stopband widths of the decimator specification as mentioned in Section 2.4.2. Therefore, this class of filters can be applied to the first stage of a multistage decimator.

Note that the $sinc^K$ filter is actually a cascade of K averaging filters and its impulse response is finite, implying it is an FIR filter. In addition, all of its impulse response coefficients are symmetric (in fact, they are all equal to unity so that no multiplications need to be implemented in hardware), and thus it is also a linear-phase filter. A $sinc^4$ filter design will be presented in Chapter 4.

Half-Band Filters

Half-band filters are characterized by the constraints that their passband and stopband ripples are the same (i.e., $\delta_p = \delta_s$) and that the cutoff frequencies are symmetrical around $\pi/2$ such that

$$\omega_p + \omega_s = \pi \tag{2.25}$$

This class of filters exhibits odd symmetry around $\pi/2$ and has zero impulse response $h(n)$ for all even values of n except $n = 0$, i.e., almost half of the filter's coefficients are zero. Therefore, these filters can be implemented with half the number of multiplications than arbitrary choices of FIR filter designs. They are appropriate for sampling rate conversion ratios of $2:1$ [22].

From Equation (2.20), we may see that if half-band filters are used as the intermediate decimators, aliasing will occur in the final transition band from F_{pc} to

F_{sc}. This transition band aliasing is a very useful and important technique that leads to a large savings in filter complexity, especially in the last stage of the decimator, as we will see from the design in Chapter 4.

Chapter 3

ANALOG MODULATOR DESIGN

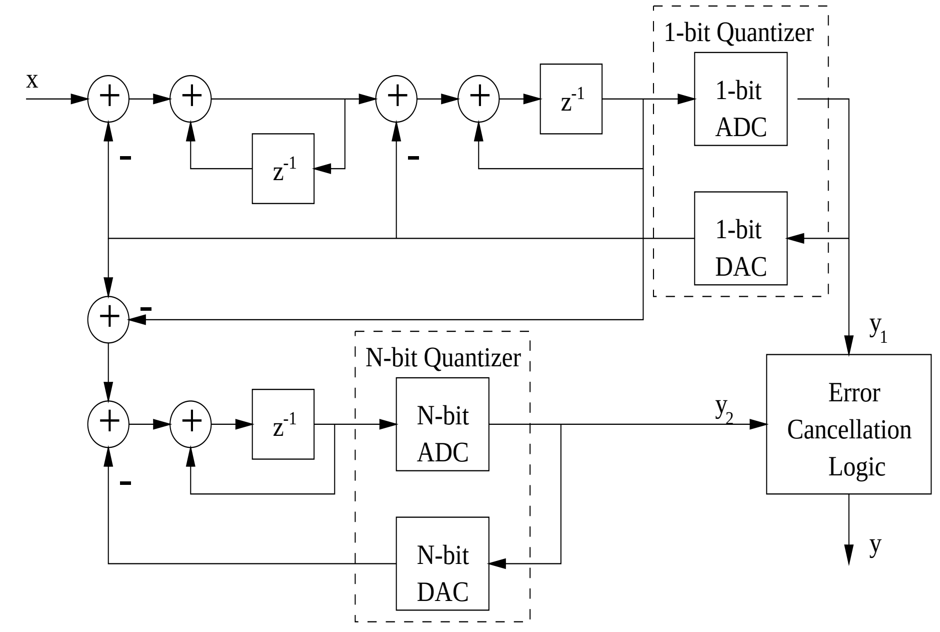
In commercial applications, oversampling $\Delta\Sigma$ converters have been successfully used for high-resolution signal acquisition in voice-band, digital audio, and ISDN fields, which have signal bands less than 100 KHz. The next significant telecommunication application for $\Delta\Sigma$ converters will be digital video with bandwidth less than 10 MHz. It is obvious that the oversampling ratio in the range of 64 to 512, typically used in oversampling converters, is impractical for such high-speed application. Alternative modulator structures are needed, working at low oversampling ratios to achieve moderate 10-12 bit resolution while maintaining the relative simplicity of analog circuit design. Dr. Brian Brandt [1] [3, Chapter 7] has implemented such a structure for 12-b 2.1 MHz conversion with $OSR = 24$ in a 1-μm CMOS technology, which is so far the fastest $\Delta\Sigma$ ADC. In my thesis work, an equivalent modulator is re-designed in a 0.8-μm CMOS technology with a few modifications of the analog circuits.

3.1 A Cascaded Multibit $\Delta\Sigma$ Modulator

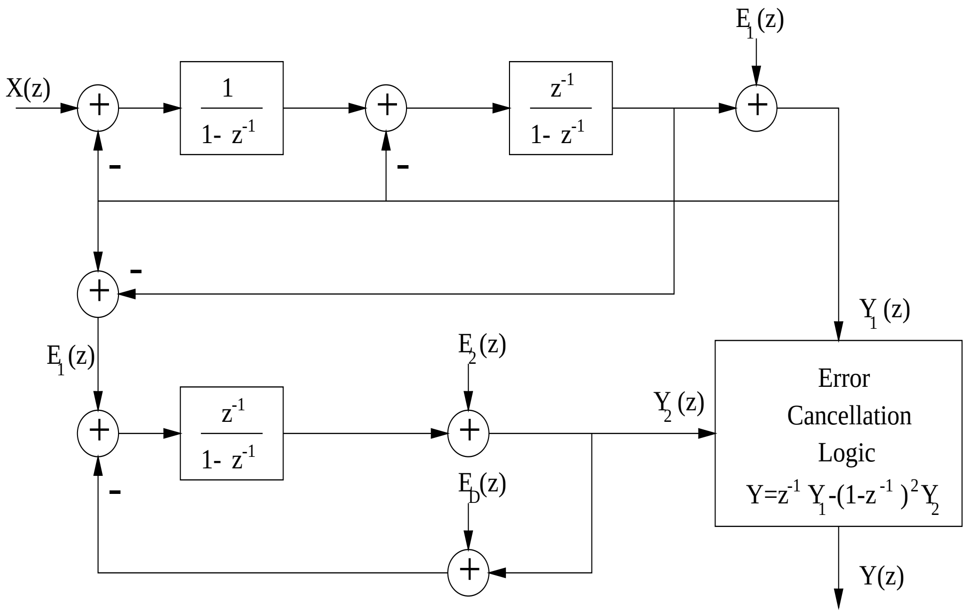
The goal of this modulator design is to reduce the oversampling ratio to as low as 16 while maintaining a dynamic range of 74 dB or, equivalently, a resolution of 12 bits, for conversion rates above 1 MHz.

3.1.1 $\Delta\Sigma$ Modulation at Low Oversampling Ratios

The dynamic range of $\Delta\Sigma$ modulator employing pure differentiation noise transfer functions depends on the oversampling ratio OSR, the order of noise shaping



(a)



(b)

Figure 3.1: Block diagram of a third-order cascaded multibit $\Delta\Sigma$ A/D converter.

L, and the internal quantizer resolution N, according to [23]

$$DR = \frac{3}{2}(\frac{2L+1}{\pi^{2L}})(2^N - 1)^2 OSR^{2L+1} \tag{3.1}$$

At a given OSR, the dynamic range may be extended by increasing L or by increasing N. Due to the term OSR^{2L+1} in Equation (3.1), the effectiveness of increasing the order of noise shaping is significantly diminished as the oversampling ratio is reduced. In contrast, the effectiveness of increasing the quantizer resolution is independent of the oversampling ratio and is therefore particularly attractive for applications at low oversampling ratios. For example, when a 4-bit quantizer is used instead of a 1-bit quantizer, the dynamic range increases nearly $20\log(2^4 - 1) \cong 24$ dB.

However, as mentioned in Section 2.3.3, linearity and resolution of a modulator based on a multibit quantizer are limited by the precision of the multibit D/A converter. To reduce this dependence on DAC linearity, several methods [3, Chapter 8], including element swapping and digital calibration, have been proposed. An alternative method is described here.

3.1.2 Linear Analysis of the Modulator

The cascaded multibit modulator shown in Figure 3.1(a) avoids sensitivity to the DAC precision by placing the multibit quantizer in the final stage of a third-order cascaded modulator. The more critical first-stage quantizer has only two analog output levels and is therefore inherently linear. The modulator consists of a second-order stage with a 1-bit quantizer followed by a first-order stage with a multibit quantizer. The input to the second stage is the difference between the output and the input of the first-stage quantizer, i.e., the quantization error of the first stage.

Shown in Figure 3.1(b) is a linear approximation of this modulator, wherein the quantizers are modeled by signal-independent additive error sources, while the integrators are represented by their transfer functions in the z-domain. The z-transform of the output of the first stage is

$$Y_1(z) = z^{-1}X(z) + (1 - z^{-1})^2 E_1(z) \tag{3.2}$$

where $E_1(z)$ models the quantization noise of the 1-bit quantizer. The input to the second stage is $E_1(z)$ and the transform of the second-stage output is

$$Y_2(z) = z^{-1}[E_1(z) - E_D(z)] + (1 - z^{-1})E_2(z) \tag{3.3}$$

where $E_2(z)$ models the quantization noise of the N-bit quantizer and $E_D(z)$ models the nonlinearity error of the N-bit DAC. The digital error cancellation logic combines the outputs from the two stages according to

$$Y(z) = z^{-1}Y_1(z) - (1 - z^{-1})^2 Y_2(z) \tag{3.4}$$

so as to cancel the quantization error $E_1(z)$ of the first stage in the final output:

$$Y(z) = z^{-2}X(z) + z^{-1}(1 - z^{-1})^2 E_D(z) - (1 - z^{-1})^3 E_2(z) \tag{3.5}$$

Thus, ideally the quantization error of the first stage is canceled and the quantization error of the second stage is attenuated in the passband by third-order shaping. Also, because $E_2(z)$ originates from a multibit quantizer, the dynamic range is improved according to Equation (3.1).

More importantly, the nonlinearity error $E_D(z)$ is attenuated in the passband by second-order shaping so that the cascaded multibit modulator much more tolerant of DAC nonlinearity than the single-stage modulator. Simulation shows that, to achieve 12-bit dynamic range, this modulator requires a DAC accuracy of only 6 bits (i.e., about 1.5% accuracy).

3.2 Modified Modulator with Interstage Coupling

In practical design of the cascaded modulator, some subunity gain factors have to be added for the integrators in Figure 3.1 in order to avoid overload at

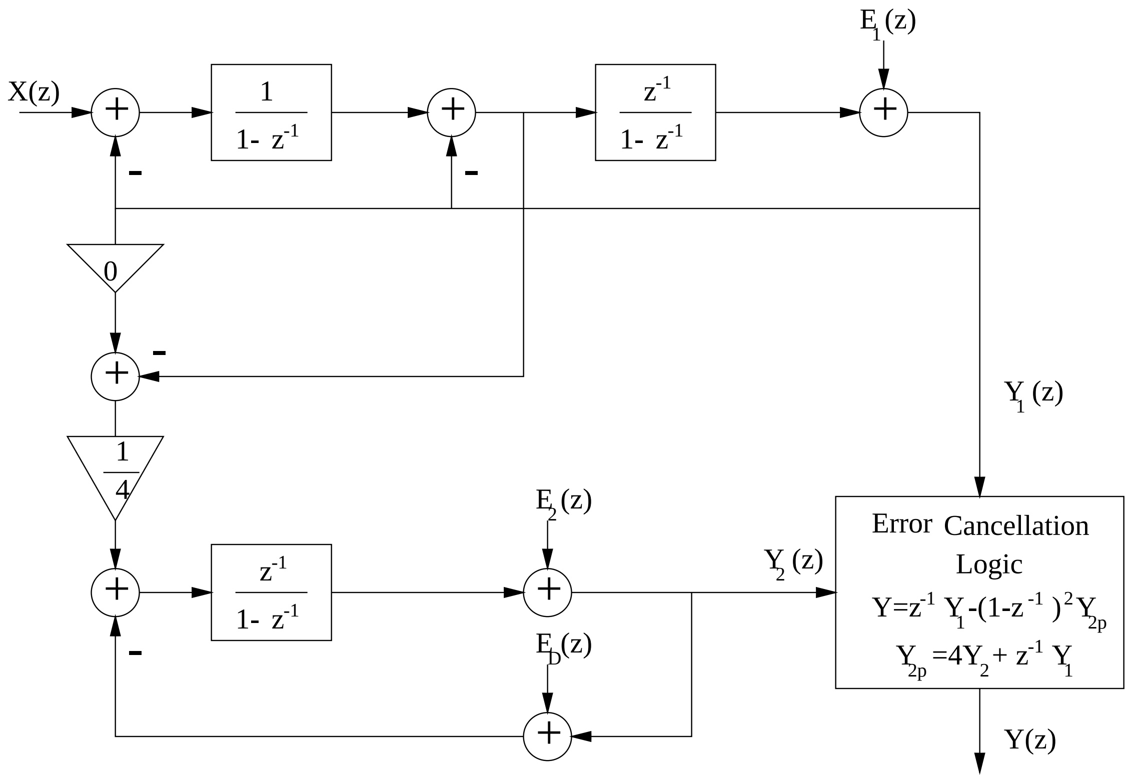


Figure 3.2: A cascaded multibit $\Delta\Sigma$ ADC with interstage coupling.

the second stage and maintain largest signal range allowed by the power supplies. Also, trade-offs have to be considered to determine the resolution N of the multibit quantizer and simplify the circuit design. Brandt has experimented several alternative choices and concluded that the measured difference in dynamic range among them was negligible [1]. Therefore, the choice with the simplest circuit design is used in this thesis work.

3.2.1 Interstage Coupling

It is preferable that both stages in the modulator have the same input range, which is defined by the two levels of the 1-bit DAC or by the two outermost levels of the N-bit DAC, in order to utilize the maximum signal swing allowed by the power supplies and also to reduce the number of required voltage references. If unity gains are used for the summation nodes in Figure 3.1(b), however, the large amplitude of the quantization error produced by the second-order first stage would overload the input range of the second stage and therefore the input range of the first stage must be reduced.

To reduce the signal range at the input to the second stage while keeping the same input range for both stages, some kind of interstage coupling coefficients should be added around the interstage summation node. An empirical selection of the interstage coupling coefficients results in a modified version of the modulator shown in Figure 3.2. In this configuration, the interstage subtraction node is actually not needed at all due to the coefficient 0 so that the complexity of the modulator is reduced. The gain coefficient $\frac{1}{4}$ reduces the output of the second integrator to one-fourth to avoid overloading the input range of the second stage. Due to these two coefficients, the $Y_2(z)$ in Equation (3.4) must be replaced by

$$Y_{2p}(z) = 4Y_2(z) + z^{-1}Y_1(z) \tag{3.6}$$

so as to prevent the overall output $Y(z)$ from containing $E_1(z)$.

The trade-offs concerning the selection of appropriate coupling coefficients are presented in [1, 24].

3.2.2 Multibit Quantizer

In principle, the multibit quantizer resolution N can be increased indefinitely to improve the dynamic range. However, in practical implementations, increasing the resolution of the multibit quantizer reduces the second-stage quantization noise and thereby increases the sensitivity to uncanceled quantization noise from the first range.

The cancellation of $E_1(z)$ depends on the noise shaping performed in the first stage precisely matching the shaping provided by the error cancellation logic. In practical, the first-stage noise shaping deviates from $(1 - z^{-1})^2$ because of circuit nonidealities such as gain errors from capacitor mismatch and dc gain in the opamps so that $E_1(z)$ cannot be totally canceled. Thus, increasing the resolution N of the multibit quantizer in the second stage increases the sensitivity to gain errors. At increasing levels of gain error, the performance of the modulator becomes dominated by uncanceled first-stage noise, and less benefit is derived from increasing the second-stage quantizer resolution.

The resolution can be tailored to the expected capacitor matching of the fabrication process. For a moderate 2% gain error margin, the benefit of a 4-bit quantizer over a 3-bit quantizer does not justify doubling the size of the second-stage quantizer. Hence, a 3-bit quantizer was chosen for the second stage.

3.2.3 Modified Integrators

Figure 3.3 shows the approximate structure of the practical modulator, where a 3-bit quantizer is used in the second stage. Two modifications of the modulator in Figure 3.1(b) are evident in the practical structure. First, both integrators in the first stage include delays in their forward paths, as well as gain factors of one-half at their inputs. Thus, the transfer function of both first-stage integrators is

$$H(z) = \frac{1}{2}\frac{z^{-1}}{1 - z^{-1}} \tag{3.7}$$

An extraction of the modified architecture from the ideal structure results in a configuration with an attenuation of 0.5 preceding the first integrator and a gain of 2 at the input of the second integrator [25]. However, since the second integrator is followed immediately by a single-threshold (1-bit) quantizer, its gain can be adjusted arbitrarily without impairing the performance of the modulator. Hence, the output of the first stage given in Equation (3.2) is changed only slightly to include an additional

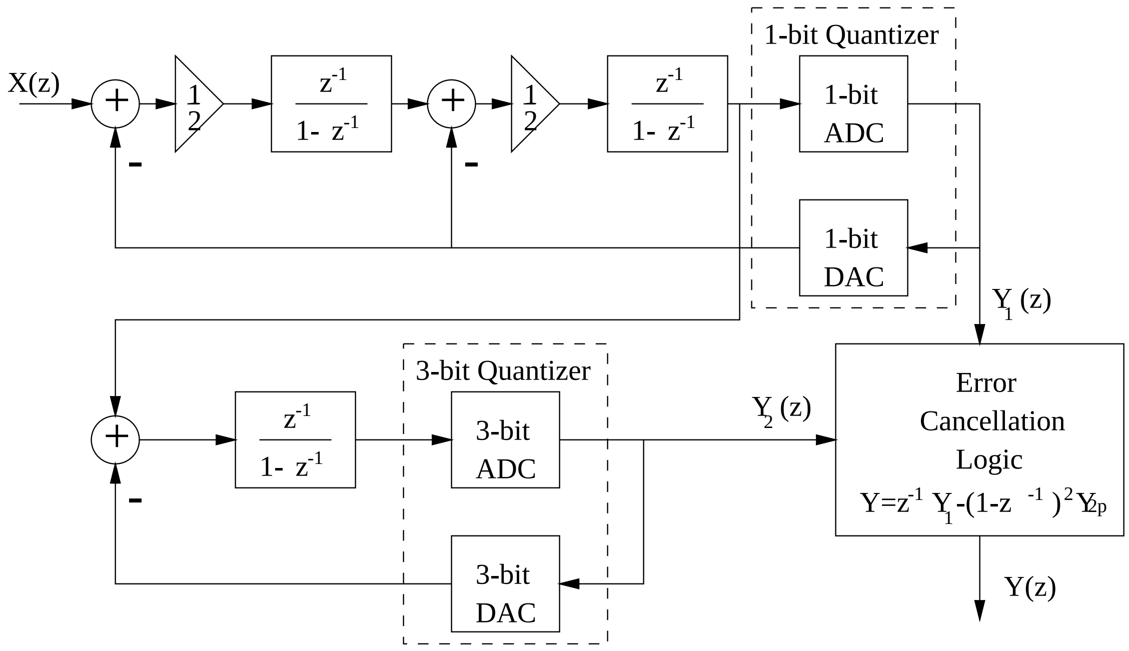


Figure 3.3: Modified structure of the third-order cascaded multibit $\Delta\Sigma$ ADC. Note that the Y_{2p} is defined in Equation (3.6).

delay (z^{-1}) preceding $X(z)$. Moreover, simulations reveal that this configuration reduces the signal range required at the outputs of the first-stage integrators to about 1.7 times the modulator's input range.

The second modification present in Figure 3.3 is that the input to the second stage is simply the output of the second integrator in the first stage. The combination of this simple interstage coupling and the $\frac{1}{2}$ gain factors in the first-stage integrators implements the configuration shown in Figure 3.2.

In practical implementations, the modulator's performance may be degraded by integrator leakage resulting from the finite dc gain of operational amplifiers, which introduces a subunity factor for z^{-1} in the denominator of Equation (3.7). Simulations show that approximately 60 dB of dc gain is required to prevent performance degradation and maintain a 12-bit dynamic range.

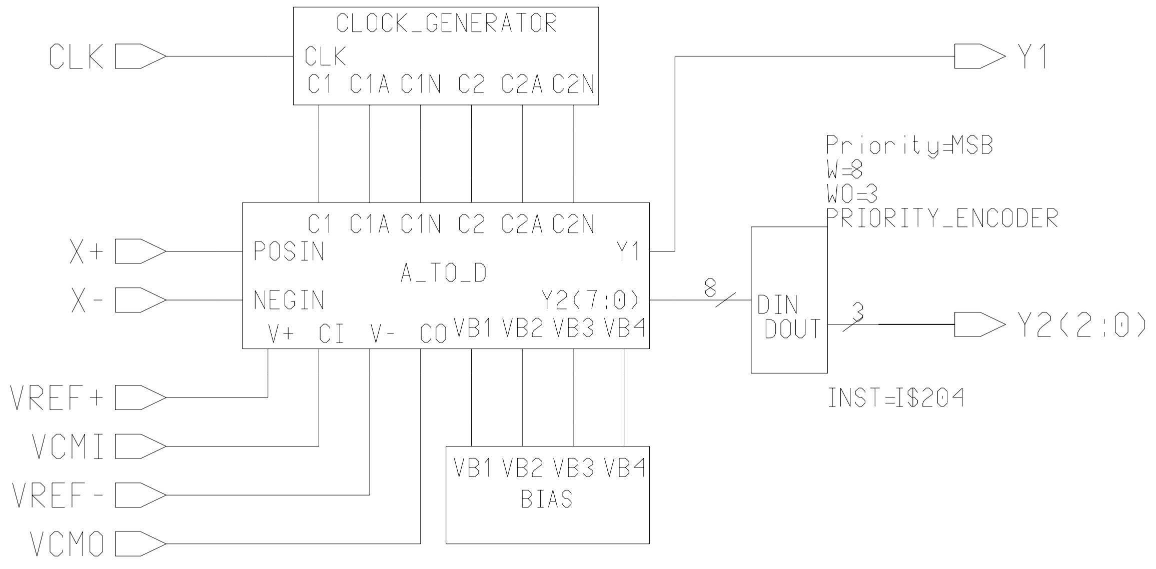


Figure 3.4: Top-level schematic of the modulator.

3.3 VLSI Implementation

The analog modulator is implemented in a 0.8-μm CMOS technology supplied by MOSIS (see Appendix A) through full-custom design methods. The performance objective is a Nyquist conversion rate of 2.1 MHz and a dynamic range of 12 bits while operating under 50 MHz clocks and from a single 5-V power supply. This section mainly discusses design issues for all of the building blocks in the modulator. All of the circuit diagrams are generated by **Design Architect**, a schematic capture tool in Mentor Graphics EDA system installed in the VLSI Lab.

3.3.1 Top-Level Structure of the Modulator

Figure 3.4 shows the topmost-level structure of the whole analog circuit and its digital interface. The inputs consist of the two differential signals X+ and X-, clock signal CLK, input reference voltages VREF+ and VREF-, common-mode input

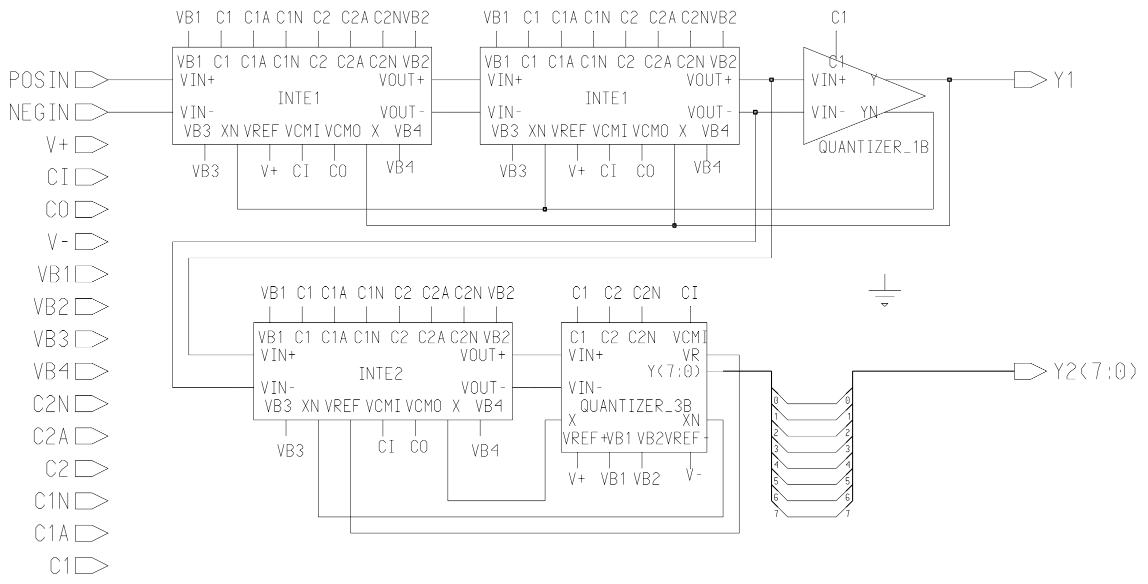


Figure 3.5: The *A-to-D* block in Figure 3.4.

voltage VCMI, and common-mode output voltage VCMO for fully differential operational amplifiers used in the *A-to-D* block. The outputs are the 1-bit quantized data Y1 and 3-bit quantized data Y2(2:0). The *BIAS* block generates four bias voltages used by opamps, and the *CLOCK_GENERATOR* block generates clocks for switched-capacitor circuits. The *PRIORITY_ENCODER* block is actually a simple 8-to-3 encoder, which is generated in VHDL (see Appendix B.1) by **LogicLib**, a VHDL library in **Design Architect**.

The *A-to-D* block has almost the same topology as that in Figure 3.3, as we may see in Figure 3.5. The first stage in Figure 3.5 consists of two integrators (*INTE1* blocks) with gain factors of $\frac{1}{2}$ and a comparator (*QUANTIZER_1B* block) that serves as the 1-bit ADC. The 1-bit DAC and the gain factor are actually implemented inside the *INTE1* blocks. The second stage consists of a single integrator with unity gain (*INTE2* block) and a 3-bit quantizer (*QUANTIZER_3B* block) which serves as both

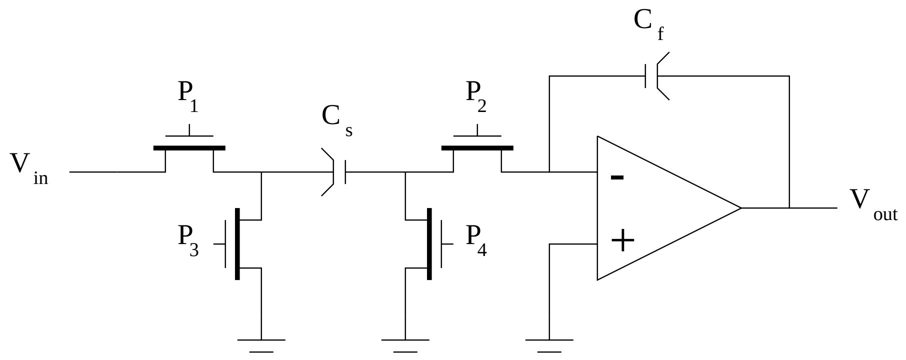


Figure 3.6: A parasitic-insensitive switched-capacitor integrator.

3-bit ADC and DAC. The *QUANTIZER_3B* block generates 8-bit data words, while the *QUANTIZER_1B* block generates complementary 1-bit outputs.

3.3.2 Switched-Capacitor Integrator

The integrators used in $\Delta\Sigma$ modulators are usually switched-capacitor (SC) circuits due to their compatibility with VLSI CMOS process as well as accurate frequency responses determined by capacitance ratios. Before analysis of the SC integrators in Figure 3.5, a brief review of SC integrator is given first.

A SC circuit is realized with the use of some basic building blocks such as opamps, capacitors, switches, and nonoverlaping clocks. Figure 3.6 shows a simple realization of a parasitic-insensitive single-ended integrator [5, Chapter 10] which consists of a single-ended opamp, a sampling capacitor C_s, a feedback capacitor C_f, and four switches controlled by clock signals P_1 to P_4. The four clock signals are grouped into two nonoverlaping clocks.

There are two categories of SC integrators. In the first case, P_1 and P_4 as well as P_2 and P_3 are identical, so the integrator has a noninverting delaying transfer function as

$$H(z) \equiv \frac{V_{out}(z)}{V_{in}(z)} = (\frac{C_s}{C_f})\frac{z^{-1}}{1 - z^{-1}} \tag{3.8}$$

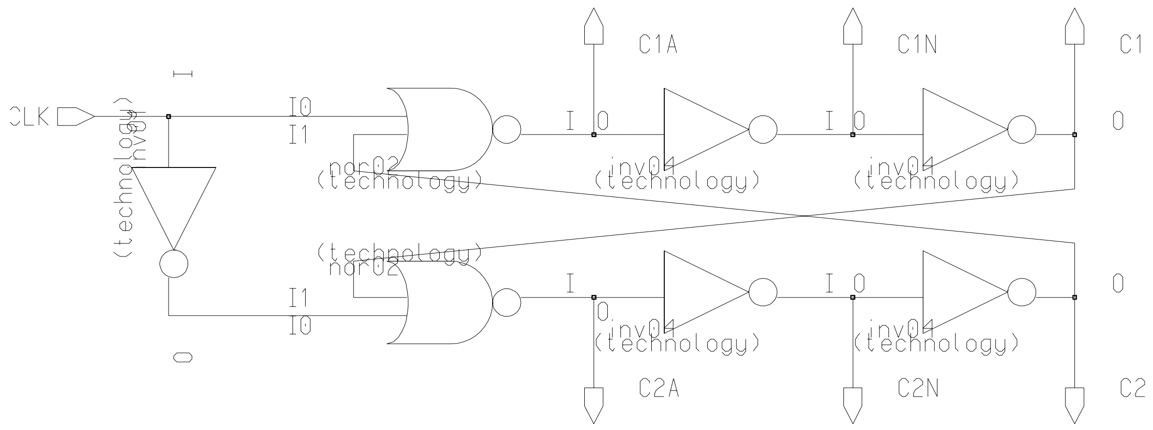


Figure 3.7: The *CLOCK_GENERATOR* block in Figure 3.4.

In the second case, P_1 and P_2 as well as P_3 and P_4 are identical, so the integrator has a inverting delay-free transfer function as

$$H(z) \equiv \frac{V_{out}(z)}{V_{in}(z)} = -(\frac{C_s}{C_f})\frac{1}{1-z^{-1}} \tag{3.9}$$

Note that the gain factors of both transfer functions are set by the ratio of C_s and C_f, which can be precisely controlled in VLSI CMOS process.

The bottom plate of the capacitors depicted in Figure 3.6 is designated by a folded line. This distinction is important because integrated capacitors are not generally symmetrical, and there is a larger parasitic capacitance to the substrate from the bottom plate than from the top plate. The capacitors should be connected such that the bottom plate is driven either directly or through a switch by a voltage source or the output of the opamp. This arrangement causes the parasitic capacitances to have the least effect on the operation of the circuit. Also, substrate noise coupling is reduced by this arrangement.

To reduce charge-injection effects in SC circuits, only n-channel switches should be used to realize the switches connected to ground or virtual ground, and clocks

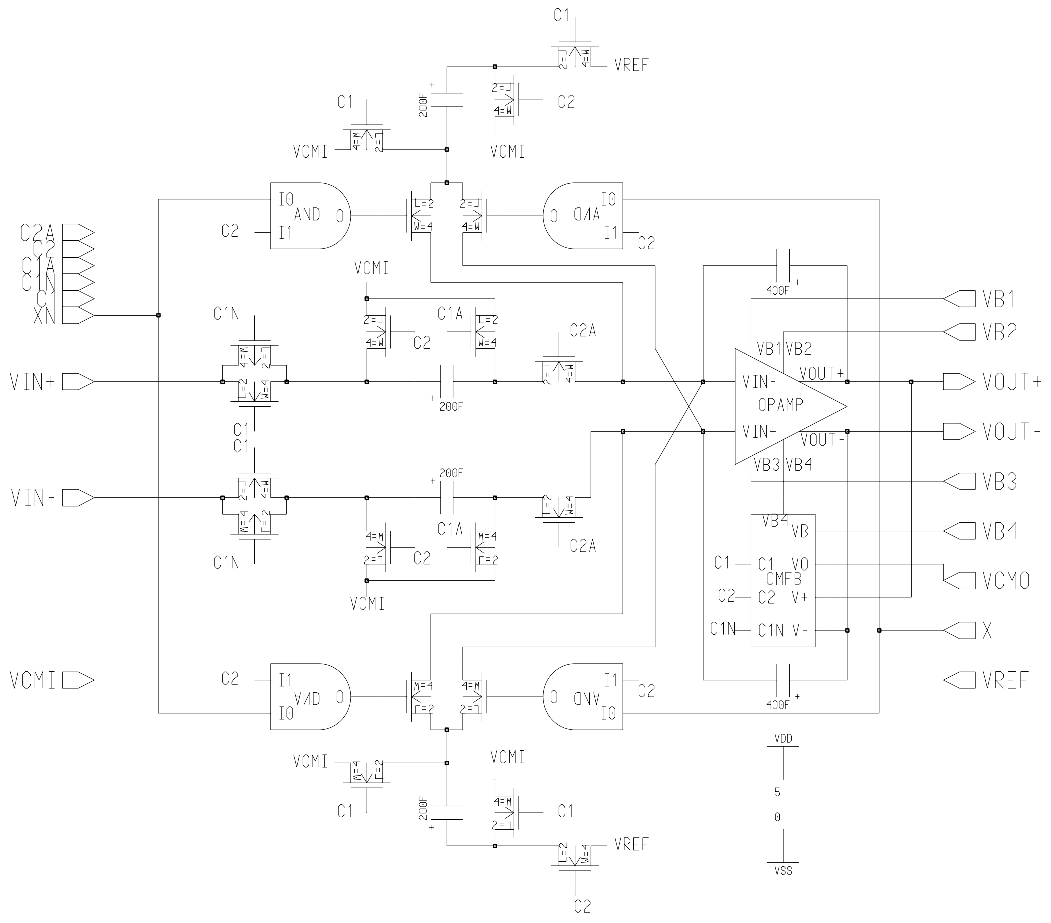


Figure 3.8: The *INTE1* block in Figure 3.5.

should be arranged so as to turn off P_2 and P_4, which are near the virtual ground node of the opamp, slightly ahead of their counterparts. A simple scheme [26] is used to generate such a two-phase nonoverlaping clock consisting of a sampling phase and an integration phase, as shown in Figure 3.7. The outputs C1 and C2 are a pair of nonoverlaping clocks, while the outputs C1A and C2A will be advanced slightly (by two inverter delays), compared to C1 and C2, respectively. The outputs C1N and C2N are complementary parts of C1 and C2, respectively.

Now we are ready to present the integrator design at a transistor level as shown in Figure 3.8. This integrator [27] has a gain factor of 0.5 defined by the ratio of sampling capacitance $200fF$ and feedback capacitance $400fF$. The only difference between the *INTE2* and *INTE1* blocks is that the feedback capacitance in *INTE2* is also $200fF$. Note that the bottom plate of the capacitors is denoted with a $+$ symbol.

In Figure 3.8, fully differential circuitry is used because it has superior power supply noise rejection, as compared to a single-ended design, and also provides twice the output swing for a given supply voltage [3, Chapter 11]. The *OPAMP* block and *CMFB* block construct a fully differential operational amplifier, which will be discussed later. The common-mode output voltage of the opamp is determined by VCMO, while the common-mode input voltage is defined by VCMI. These two voltages can be different.

All of the switches, except for the two switches directly connected with inputs VIN+ and VIN-, are realized by minimum-gate-length NMOS FETs ($W/L = 1.6\mu m/0.8\mu m$) so as to minimize parasitics, while maintaining good conduction for low enough analog ground VCMI. However, full CMOS transmission gates are used for the switches coupled to the signal input, since they must conduct over a wide range of voltages.

The *INTE1* block realizes both an integrator and a 1-bit DAC. To understand this, first consider only the signal path from VIN+ to the input of the *OPAMP* block. This signal path has the same configuration as that shown in Figure 3.6 in the case that P_1 and P_4 correspond to the identical clocks C1 and C1A while P_2 and P_3 correspond to the identical clocks C2 and C2A. Therefore, the transfer function of this integrator is defined by Equation (3.8).

Recalling from the previous discussion for Equation (3.8) and (3.9), it is easy to see that the upper VREF path is noninverting while the lower VREF path is inverting, which may correspond to positive VREF and negative VREF, respectively. During phase C1, the top capacitor charges to VREF while the lower capacitor charges to ground. During phase C2, the inputs X and XN shown in Figure 3.8, which are complementary digital outputs of a comparator (i.e., a 1-bit ADC), decide which summing junction is to receive a positive reference charge and which is to receive a negative reference charge. If X is high (and XN is low), then a reference charge is taken from the lower summing junction, while a reference charge is delivered to the upper summing junction. If X is low, the charge delivery is swapping. This circuit realizes the 1-bit DAC while avoiding modulation of the reference voltage [3, Chapter 11].

3.3.3 Wide-Swing Constant-Transconductance Bias Circuit

The first problem that I encountered in opamp design was to figure out proper bias voltages to make the opamp work in the region where moderate gain can be achieved. In Brandt's original design, no bias circuitry was designed and conventional cascode current mirrors were used. In this thesis work, a constant-transconductance bias circuit is designed exploiting wide-swing cascode current mirrors [5, Chapter 6], as shown in Figure 3.9.

A conventional cascode current mirror is shown in Figure 3.10(a). Although its output impedance is increased to $r_{ds4}(r_{ds2}g_{m4})$, where g_{m4} is the transconductance of the transistor Q_4 and r_{ds} is drain-source resistance of a transistor, a cascode current mirror reduces the maximum output-signal swing so that the minimum allowed voltage for V_{out} is V_{tn} (threshold voltage of NMOS FET, approximately 0.7V) greater than $2V_{eff}$, where V_{eff} is the minimum drain-source voltage (typically around 0.2–0.25V) needed to keep a transistor working in the saturation region. This loss of signal swing is a serious disadvantage for modern VLSI technologies.

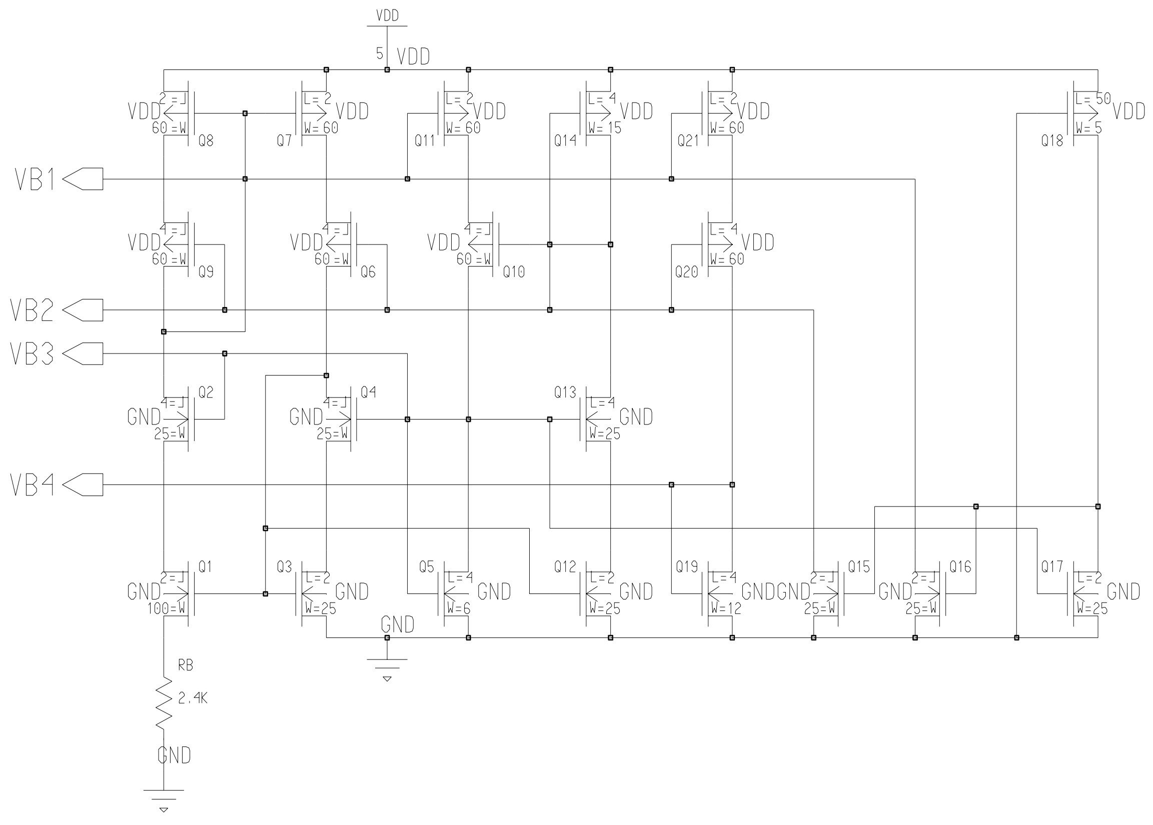


Figure 3.9: The *BIAS* block in Figure 3.4.

An alternative circuit that does not reduce the signal swing so much while keeping high output impedance, often called the *wide-swing cascode current mirror*, is shown in Figure 3.10(b). The minimum allowable output voltage just needs to be greater than $(n+1)V_{eff}$. When selecting n as unity, this current mirror can guarantee that all of the transistors are in the saturation region even when V_{out} drops to as small as 0.4–0.5V. In practical designs, the lengths of Q_2 and Q_3 should be minimized in order to maximize the frequency response, as their gate-source capacitances are the most significant capacitances contributing to high-frequency poles. However, Q_1 and Q_4 should be chosen to have longer gate lengths, typically twice the minimum channel length, to eliminate detrimental short-channel effects. Also, the aspect ratio of Q_5,

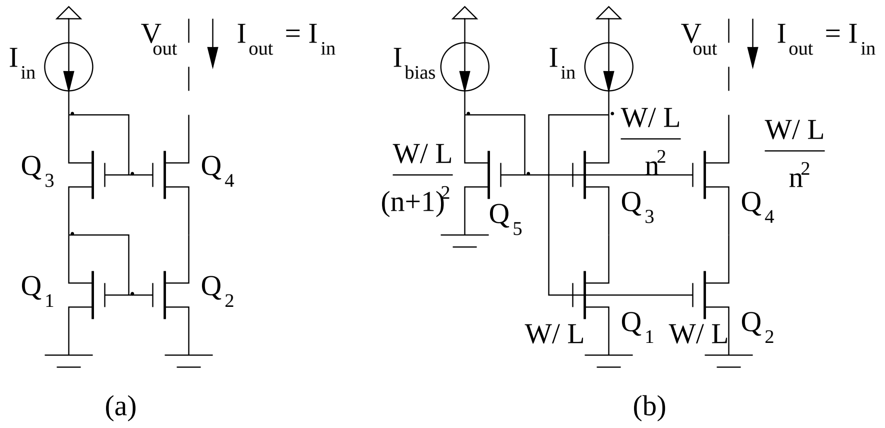


Figure 3.10: Conventional (a) and wide-swing (b) cascode current mirrors. I_{bias} typically is set to the nominal or maximum input current, I_{in}.

$(W/L)_5$, should be smaller than the size given in Figure 3.10(b) in order to offset the body effect of Q_1 and Q_4. All of these practical considerations have been taken into account in the bias circuit design shown in Figure 3.9.

In Figure 3.9, the width and length of all the transistors are denoted in units of *lambda*. For instance, Q_1 has length of 4 lambda and width of 25 lambda. Table 3.1 lists the W/L ratios ($\mu m/\mu m$) of all the transistors for a MOSIS $0.8-\mu m$ process. The n-channel wide-swing cascode current mirror consists of transistors Q_1–Q_4, along with the diode-connected biasing transistor Q_5. The output current comes from Q_1, and the bias current is actually derived from the bias loop via Q_{10} and Q_{11}. Similarly, the p-channel wide-swing cascode current mirror is realized by Q_6–Q_9, along with diode-connected Q_{14} which has a bias current derived from the bias loop via Q_{12} and Q_{13}. The output current is the drain current of Q_6.

Transistor transconductances are perhaps the most important parameters in opamps that must be stabilized. This stabilization can be achieved by using resistor R_B and setting $(W/L)_2 = 4(W/L)_3$ as shown in Figure 3.9, where transistor transconductances are matched to the conductance of R_B. Specifically, the transconductance

Table 3.1: Transistors in the Bias Circuit.

Transistor	$W/L(\mu m/\mu m)$
Q_8, Q_7, Q_{11}, Q_{21}	24/0.8
Q_9, Q_6, Q_{10}, Q_{20}	24/1.6
$Q_3, Q_{12}, Q_{15}, Q_{16}, Q_{17}$	10/0.8
Q_1, Q_4, Q_{13}	10/1.6
Q_2	40/0.8
Q_{14}	6/1.6
Q_5	2.4/1.6
Q_{19}	4.8/1.6
Q_{18}	2/20

of Q_3 is set to

$$g_{m3} = \frac{1}{R_B} \tag{3.10}$$

Since all transistor currents (including those in opamps) are derived from the same biasing network and the ratios of the currents are mainly dependent on geometry, all other transconductances are also stabilized so that for all n-channel transistors,

$$g_{mi} = \sqrt{\frac{(W/L)_i I_{Di}}{(W/L)_3 I_{D3}}} \times g_{m3} \tag{3.11}$$

and for all p-channel transistors

$$g_{mi} = \sqrt{\frac{\mu_p}{\mu_n} \frac{(W/L)_i I_{Di}}{(W/L)_3 I_{D3}}} \times g_{m3} \tag{3.12}$$

Therefore, this kind of bias circuits is called *constant-transconductance bias circuit.*

The bias loop does have the problem that at start-up it is possible for the current to be zero in all transistors, and the circuit will remain in this stable state forever. Thus, it is necessary to include the start-up circuitry consisting of Q_{15}–Q_{18} to inject currents into the bias-loop in the case that all currents in the loop are zero. Once the loop starts up, this start-up circuitry is disabled via turning off Q_{15} and

Q_{16}. Note that Q_{18} has much higher impedance than Q_{17} when the latter is turned on.

A simple current mirror consisting of Q_{19}–Q_{21} is added to supply a bias voltage for setting up the common-mode output level of opamps.

3.3.4 Fully Differential Folded-Cascode Opamp

The operational amplifier used in the integrators is the most critical element of the modulator. The design of a proper opamp involves a great amount of theoretical calculations and empirical simulations to achieve fast speed and sufficient gain. In $\Delta\Sigma$ modulators, if the opamp is fast enough (i.e., it has a small enough settling time constant $\tau = 1/(\beta\omega_t)$, where β is the feedback factor and ω_t is the unity-gain frequency of the opamp), small sampling period T (i.e., high sampling frequency and oversampling ratio) may be used while allowing the circuit to settle completely. Usually, the unity-gain frequency of the opamp should be at least five times greater than the sampling frequency (e.g., $T > 5\tau$) to ensure the gain error introduced by incomplete settling of the integrator less than $\epsilon^{-T/\tau} = \epsilon^{-5\tau/\tau} \simeq 0.7\%$. The amplifier should also have a reasonable amount of low-frequency open-loop gain, so the distortion of the amplifier will be reduced. Since opamps in a $\Delta\Sigma$ modulator usually drive only capacitive loads, it is possible to realize an opamp with only a single high-impedance node at the output while having all internal nodes of relatively low impedance, in order to maximize the speed.

Given these design criteria, a good choice for the opamp in the switched-capacitor integrators of a $\Delta\Sigma$ modulator is a fully differential folded-cascode opamp as shown in Figure 3.11 [28]. The aspect ratios of all transistors in the opamp are listed in Table 3.2. The basic idea of the folded-cascode opamp is to apply cascode transistors to the input differential pair but using transistors opposite in type from those used in the input stage. For example, the differential-pair transistors consisting

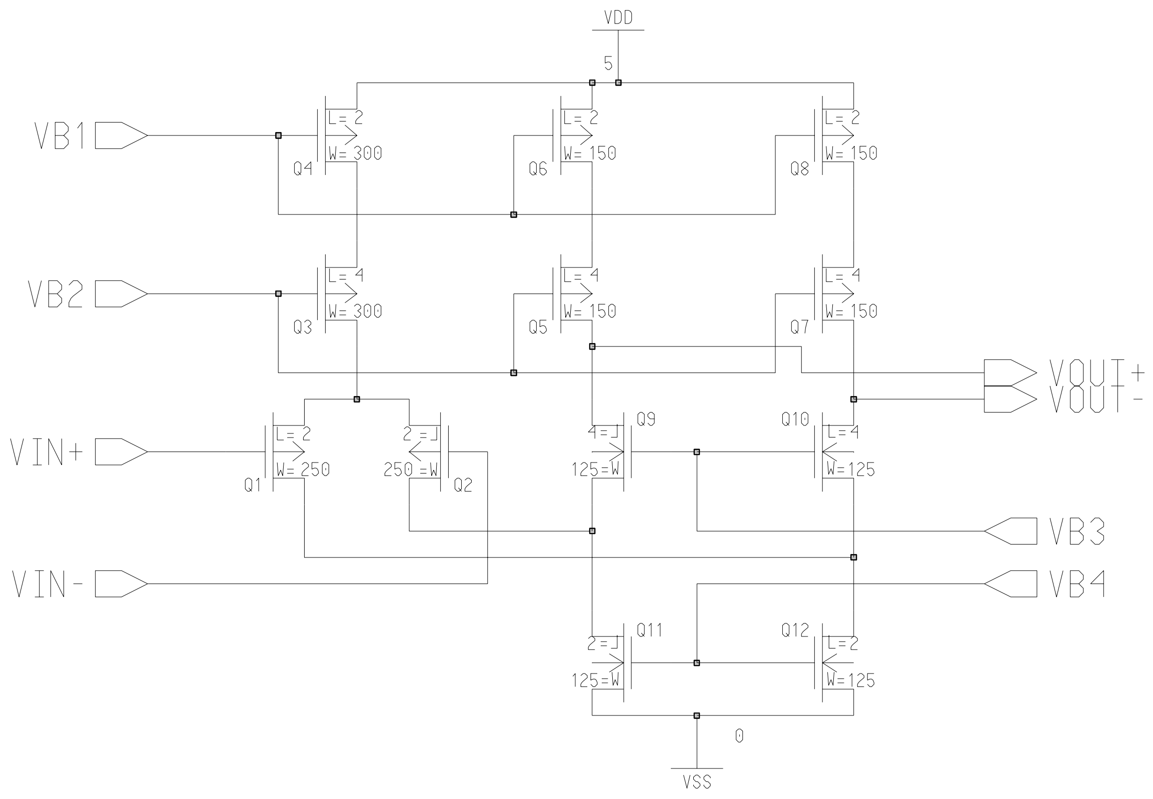


Figure 3.11: The *OPAMP* block in Figure 3.8.

of Q_1 and Q_2 are p-channel transistors in Figure 3.11, whereas the cascode transistors consisting of Q_9 and Q_{10} are n-channel transistors, which are biased by VB3 generated by the bias circuit shown in Figure 3.9. The p-channel transistors Q_3–Q_8 construct three cascode current sources biased by VB1 and VB2 from the bias circuit to provide bias currents and active loads. Note that PMOS wide-swing cascode current mirrors are used here and the common-mode currents are set as $I_{D1} = I_{D2} = I_{D9} = I_{D10}$.

The dc gain of this opamp is determined by the product of the input transconductance g_{m1} and the output impedance r_o (i.e., looking into the drains of Q_7 and Q_{10}), and r_o is quite high due to the use of cascode techniques. A small-signal analysis

Table 3.2: Transistors in the Opamp Circuit.

Transistor	$W/L(\mu m/\mu m)$
Q_4	120/0.8
Q_3	120/1.6
Q_6, Q_8	60/0.8
Q_5, Q_7	60/1.6
Q_1, Q_2	100/0.8
Q_9, Q_{10}	50/1.6
Q_{11}, Q_{12}	50/0.8

gives the dc gain as

$$A_v = -g_{m1} r_o = -g_{m1}[g_{m10} r_{ds10}(r_{ds12} \parallel r_{ds1}) \parallel g_{m7} r_{ds7} r_{ds8}] \tag{3.13}$$

where transconductances g_{mi} can be estimated from Equation (3.11) and (3.12) and drain-source resistances r_{dsi} are typically in the order of $10^2 k\Omega$. The cascode opamp is compensated by its load capacitance at the output nodes (not shown in Figure 3.11). For mid-band and high frequencies, the load capacitance C_L dominates, and the unity-gain bandwidth of the amplifier is determined by the ratio of the transconductance of an input device to C_L in a single dominant pole response, i.e.,

$$\omega_t = \frac{g_{m1}}{C_L} \tag{3.14}$$

From this equation, it seems that the opamp could be made arbitrarily fast by increasing the width and bias current of the input devices to increase g_m. However, to maintain stability, all nondominant poles must occur at frequencies higher than the dominant pole. Therefore, the speed of the opamp is limited by the location of the first nondominant pole determined by the ratio of the transconductance of the cascode NMOS device to the total capacitance on its source (i.e., g_{m10}/C_{s10}).

Note that the bias voltage VB4 in Figure 3.11 does not directly come from the bias circuit shown in Figure 3.9, instead, it comes from the *common-mode feedback*

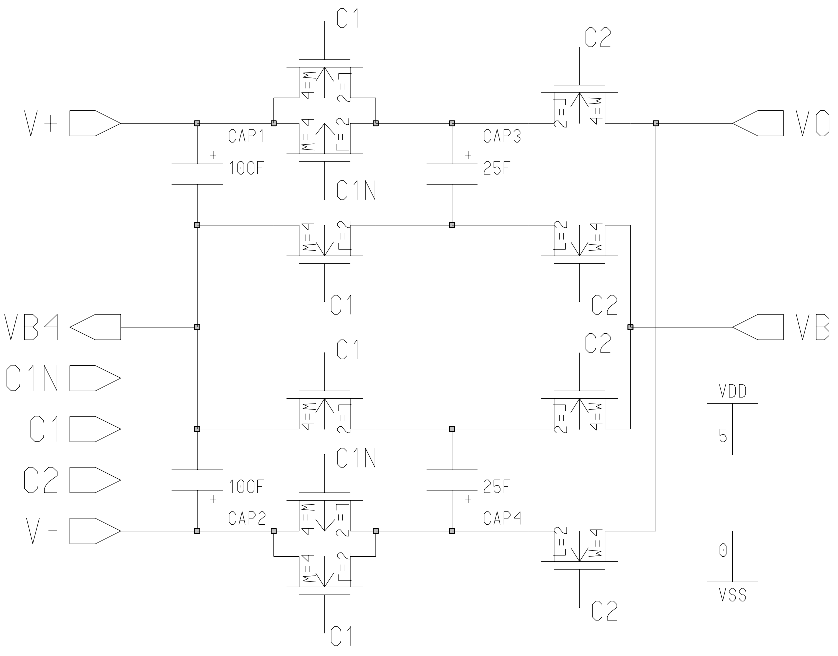


Figure 3.12: The *CMFB* block in Figure 3.8.

(CMFB) circuitry shown in Figure 3.12. One drawback of using fully differential opamps is that a CMFB circuit must be added to establish the common-mode (i.e., average) output voltage. Ideally, it will keep this common-mode voltage immovable, preferably close to halfway between the power-supply voltages, even when large differential signals are present. Without it, the common-mode voltage is left to drift, since the common-mode loop gain is not typically large enough to control its value. The performance requirements on the CMFB circuitry are not nearly as stringent as for the main opamp, because the signal of interest is the difference between the main opamp outputs.

Figure 3.12 shows a CMFB circuit only used in switched-capacitor circuits. The connections between the CMFB circuit and the opamp are depicted in Figure 3.8. It may be seen that nonswitched capacitors *CAP1* and *CAP2* form a voltage divider

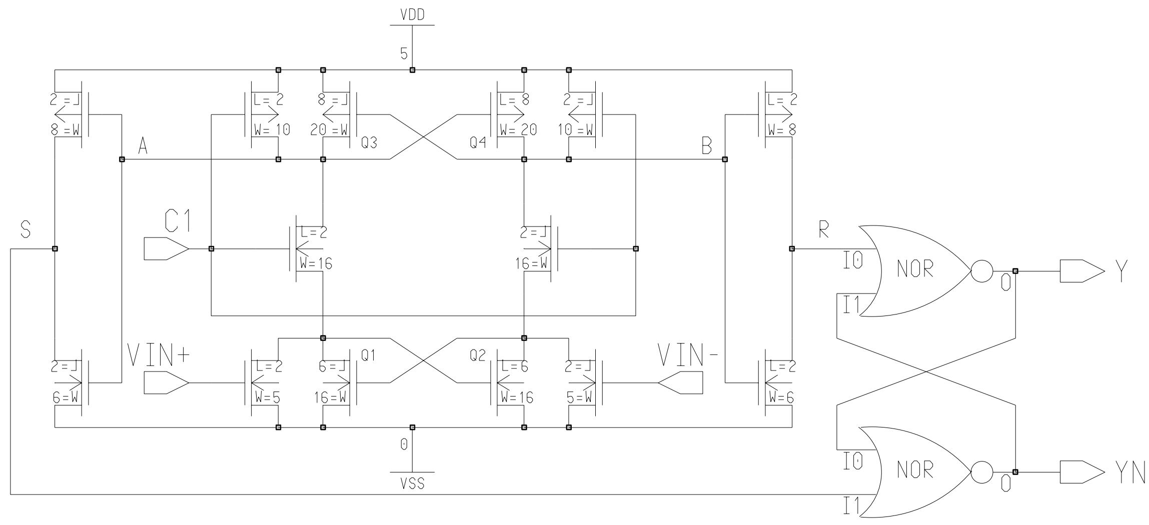


Figure 3.13: The *QUANTIZER_1B* block in Figure 3.5.

to generate the average of the opamp output voltages, and switched capacitors *CAP3* and *CAP4* transfer corrective charges onto *CAP1* and *CAP2* during phase 1 (C1). The bias voltage VB is chosen to set the common-mode output voltage of the opamp around VCMO ($2.5V$ in this case). CMOS transmission gates are used to realize the switches connected to the outputs of the opamp, in order to accommodate a wider signal swing. The switched capacitors ($25fF$) are set to be one-quarter the sizes of the nonswitched capacitors ($100fF$) so as to avoid common-mode offset voltages and opamp overload [5, Chapter 6].

3.3.5 Regenerative Track-and-Latch Comparator

The second major building block in a $\Delta\Sigma$ modulator is the comparator which quantizes a signal in the loop and provides the output of the modulator. Since the comparator appears after the loop gain block and before the output terminal, nonidealities associated with it are shaped by the loop in the same way that the quantization noise it produces is shaped. Therefore, the performance of the modulator is relatively insensitive to offset and hysteresis (i.e., the tendency that a comparator might have to stay in the previous direction when it should toggle to another direction) in the first-stage comparator. A fast regenerative latch without preamplification, as shown in Figure 3.13, has been used to implement the first-stage comparator. In this latch the cross-coupled devices Q_1–Q_2 and Q_3–Q_4, are strobed at their drains, rather than at the sources, to eliminate backgating effects and promote faster regeneration [5, Chapter 7].

During the *latch* phase the clock input C1 is low, thereby disabling the regeneration while nodes A and B are pulled high. Nodes R and S, which are the inputs to the RS latch constructed by two NOR gates, are consequently both low. When the comparator is strobed by bringing C1 high (*track* phase), nodes A and B are released and positive feedback, produced by the cross-coupling of transistors Q_1–Q_2 and Q_3–Q_4, regenerates analog input signal into a full-scale digital signal.

3.3.6 3-bit Flash ADC and 3-bit DAC

The modulator performance is also very tolerant of nonlinearity and hysteresis in the second-stage 3-bit ADC because their effects are attenuated in the baseband by third-order noise shaping. The design of this ADC is complicated by the fact that it must compare the differential output voltage of the second-stage integrator to seven differential reference voltages. Because the clock frequency is high and the comparisons should be completed in one clock cycle, a 3-bit flash A/D converter [29] is used in this implementation as shown in Figure 3.14. It should be noted that a modified 3-bit differential DAC is also included in Figure 3.14 in order to share one resistor string with the ADC. The resistor string divides the reference into equal segments as thresholds followed by a bank of comparators (i.e., *COMP* blocks). Since the outputs of the second-stage integrator (VIN+ and VIN- in Figure 3.14) need to drive seven comparators which introduce large load, source followers (i.e., *BUFF*

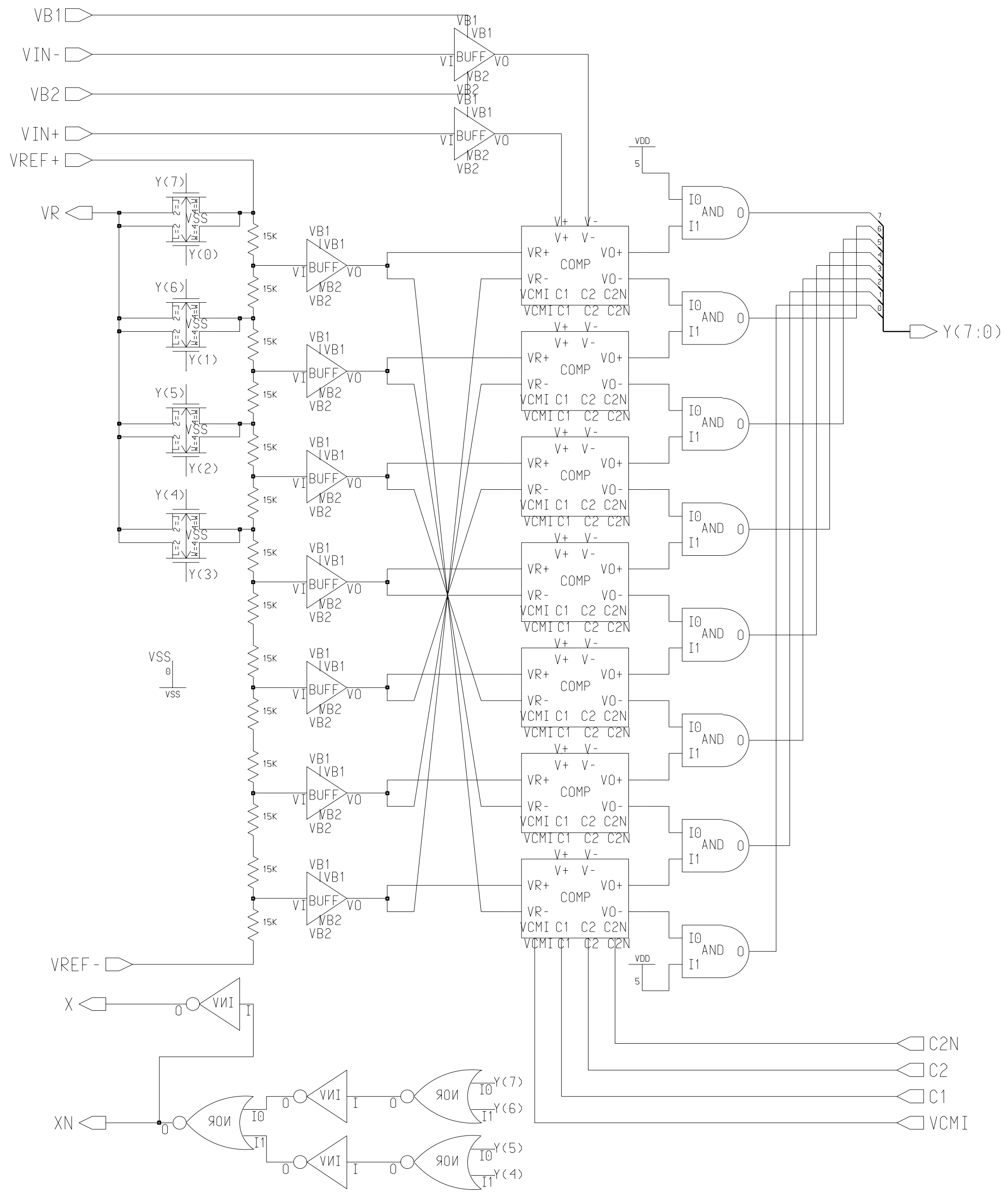


Figure 3.14: The *QUANTIZER_3B* block in Figure 3.5.

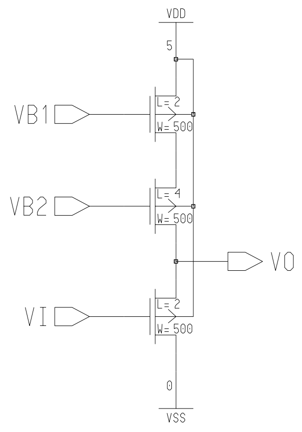


Figure 3.15: The *BUFF* block shown in Figure 3.14.

blocks) are added to buffer the loading. Equivalent buffers are placed between the resistor string and the comparator bank to compensate for gain error and nonlinearity introduced by the source followers. The source follower is shown in Figure 3.15 where the active load is biased by VB1 and VB2 from the bias circuit.

The *COMP* block is a switched-capacitor circuit combined with the regenerative track-and-latch comparator, as shown in Figure 3.16. During phase 1 (C1), the two capacitors are charged to unique differential voltages derived from the resistor string, and during phase 2 (C2), the left sides of the capacitors are driven by the outputs of the second-stage integrator. Note here that CMOS transmission gates are connected to the signals for wider swing. The comparator is strobed at the end of C2. As an overall output of the 3-bit flash ADC, a 1-out-of-8 code is produced by the AND gates in Figure 3.14.

The modulator's tolerance of nonlinearity in the second-stage 3-bit DAC permits the use of a simple differential tapped resistor string for implementation [1], as is illustrated in Figure 3.17. In order to utilize the 1-bit DAC structure shown in Figure 3.8, however, I modified a normal differential DAC by cutting off the lower

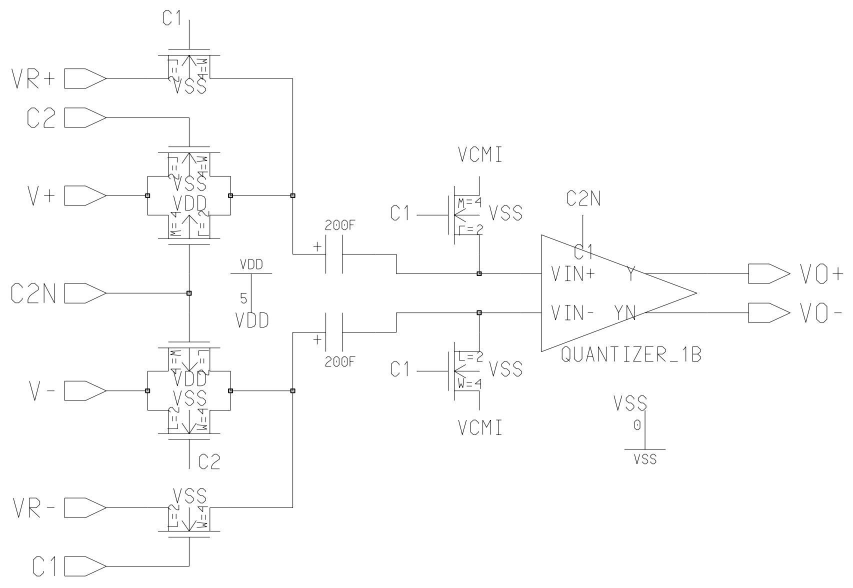


Figure 3.16: The *CMPR* block shown in Figure 3.14.

half of the DAC and adding a combinational logic to control the 1-bit DAC, as shown in the left portion of Figure 3.14. In Figure 3.17, after a pair of taps from the resistor string, centered with the average of V_{ref+} and V_{ref-}, are selected to complete the D/A conversion, the outputs Out+ and Out- are fed into the upper and lower summing junctions of the integrator, respectively. In my design, only one tap from the upper resistor string is selected and the output X decides whether VR should be delivered to or taken from the summing junctions of the integrator shown in Figure 3.8. For instance, when any of Y(4)–Y(7) in Figure 3.14 is high, X goes high and VR is taken from the lower summing junction and delivered to the upper summing junction of the integrator.

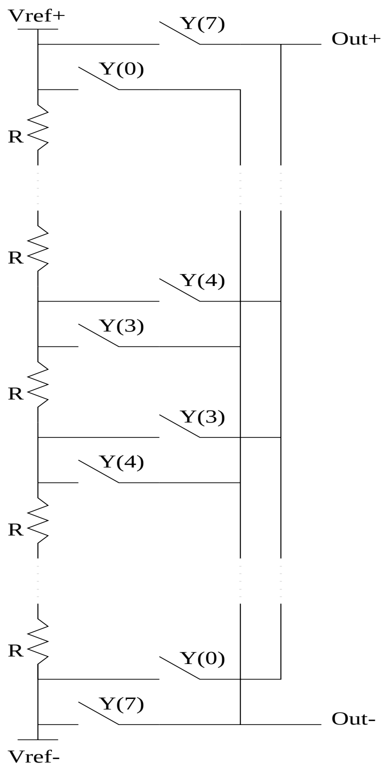


Figure 3.17: Block diagram of a 3-bit differential DAC.

3.3.7 Simulation

All of the above circuits were simulated within Mentor Graphics analog simulator, **AccuSim**, after schematic captures. Three types of analysis may be applied to the circuits in the simulator: dc, ac, and transient analysis. The simulations exploit SPICE MOSFET models (see Appendix A.2) measured from a specific process run. Since the clock generator in Figure 3.7 is composed of logic gates from a MOSIS standard cell library and the library does not have SPICE models for analog simulation, I could not simulate this digital circuit with other analog circuits at the same time, instead, I had to simulate the switched-capacitor circuits by adding arbitrary pulse signals to emulate the clocks. All of the pulse signals have same cycle of $20ns$ (i.e.,

Table 3.3: Clock Signals Defined in the Simulator.

Clocks	Pulse Delay (nanosecond)
C1, C2N	0
C2, C1N	10
C1A	19
C2A	9

frequency of 50 MHz), pulse width of $8ns$, rise and fall time of $1ns$, and pulse range of 0–5 volts. Table 3.3 lists phase differences of the six pulse signals.

Figure 3.18 shows the transient analysis results of the bias circuit in Figure 3.9 as R_B is chosen to be $2.4k\Omega$ so that the current-mirror output current is about $30\mu A$. Simulations reveal that the value of R_B is not very crucial for the stability of the circuit, and $+3\%$ derivation (5-bit accuracy) of R_B is tolerable for maintaining enough open-loop low-frequency gain of the opamp. Under these bias voltages, the opamp in Figure 3.11 has a open-loop frequency response with $200fF$ capacitive load on each of its outputs, as shown in Figure 3.19. The ac analysis of the opamp circuit shows a dc gain of 53 dB and a unity-gain frequency of 450 MHz, which are acceptable for the present application. Figure 3.20 shows the transient analysis results of the opamp circuit driven by a 1 MHz fully differential sinusoidal input of 1 millivolt centered at 2.5 volt. The outputs of the opamp have a common-mode voltage of 2.5 volt, a gain of 48 dB, a phase shift of about $40°$. The bias current flowing through Q_3 is about 143 μA, less than the ideal value of 150 μA, due to the finite output impedance of the current source constructed by Q_3 and Q_4 in Figure 3.11.

The transient analysis of the whole modulator is very time-consuming because the circuits contain around 500 devices (transistors, capacitors, and resistors). A 10-microsecond transient simulation takes more than 24 hours for a Sun SPARC 20 workstation to complete. To simulate the circuit shown in Figure 3.5, the bias circuit

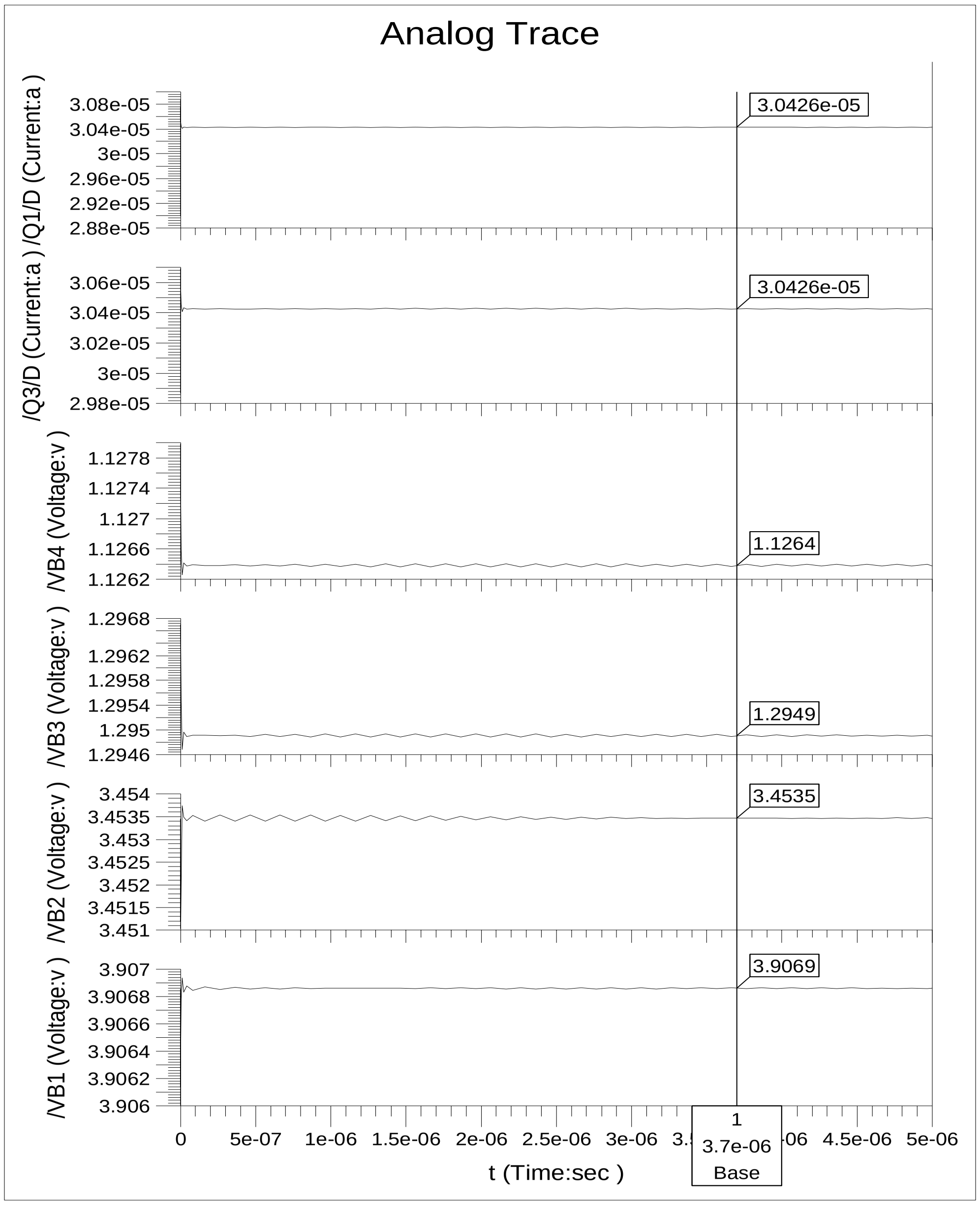


Figure 3.18: Bias voltages from simulation of the circuit shown in Figure 3.9.

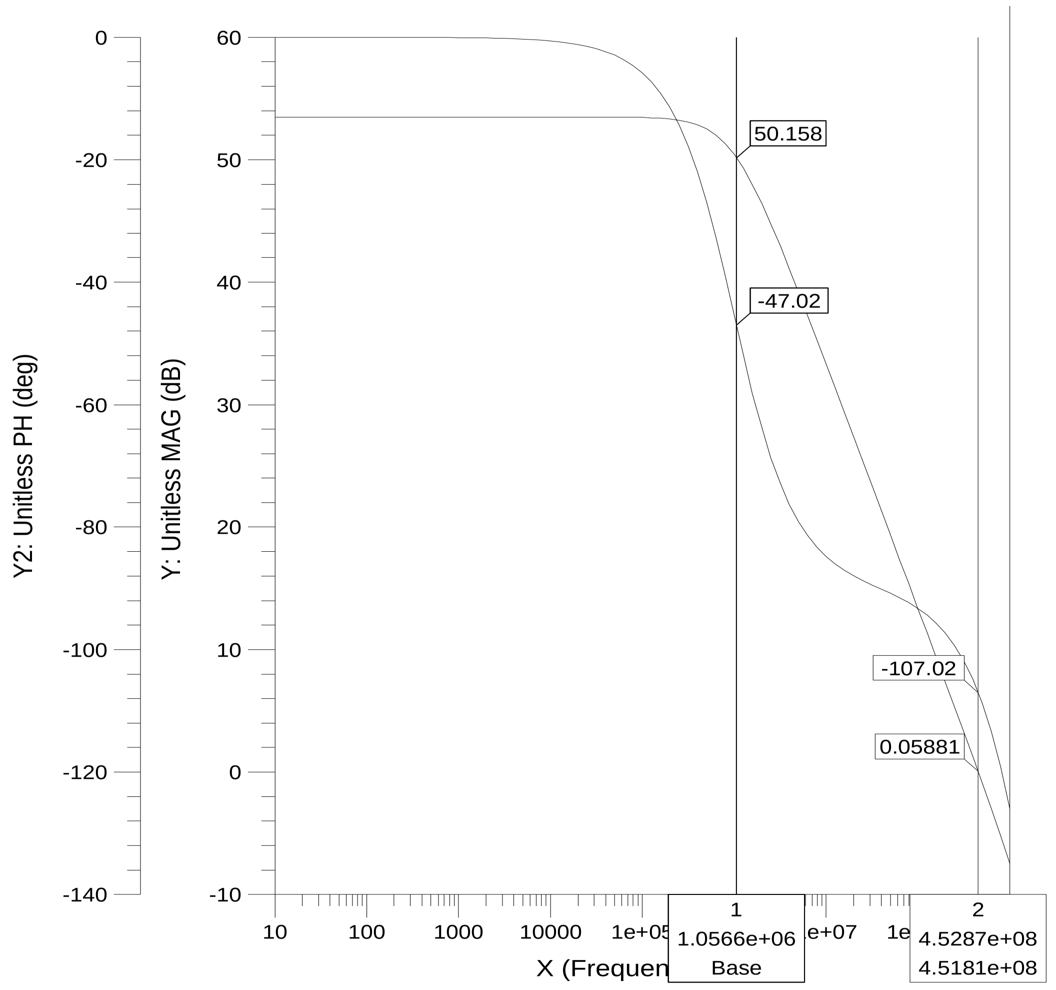


Figure 3.19: Open-loop freqency response of the opamp shown in Figure 3.11 with $200fF$ capacitive load.

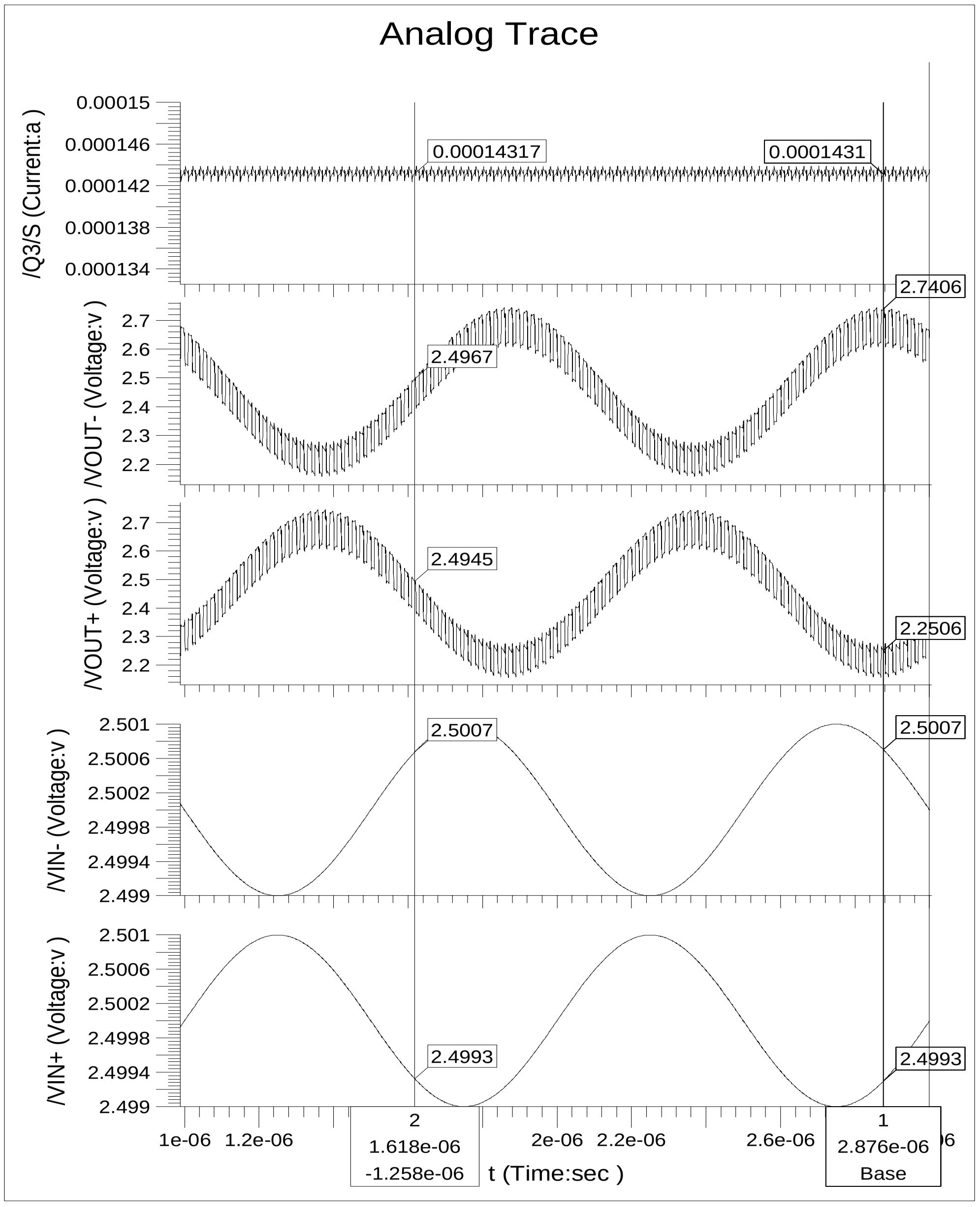


Figure 3.20: Open-loop transient analysis of the opamp shown in Figure 3.11 with $200fF$ capacitive load.

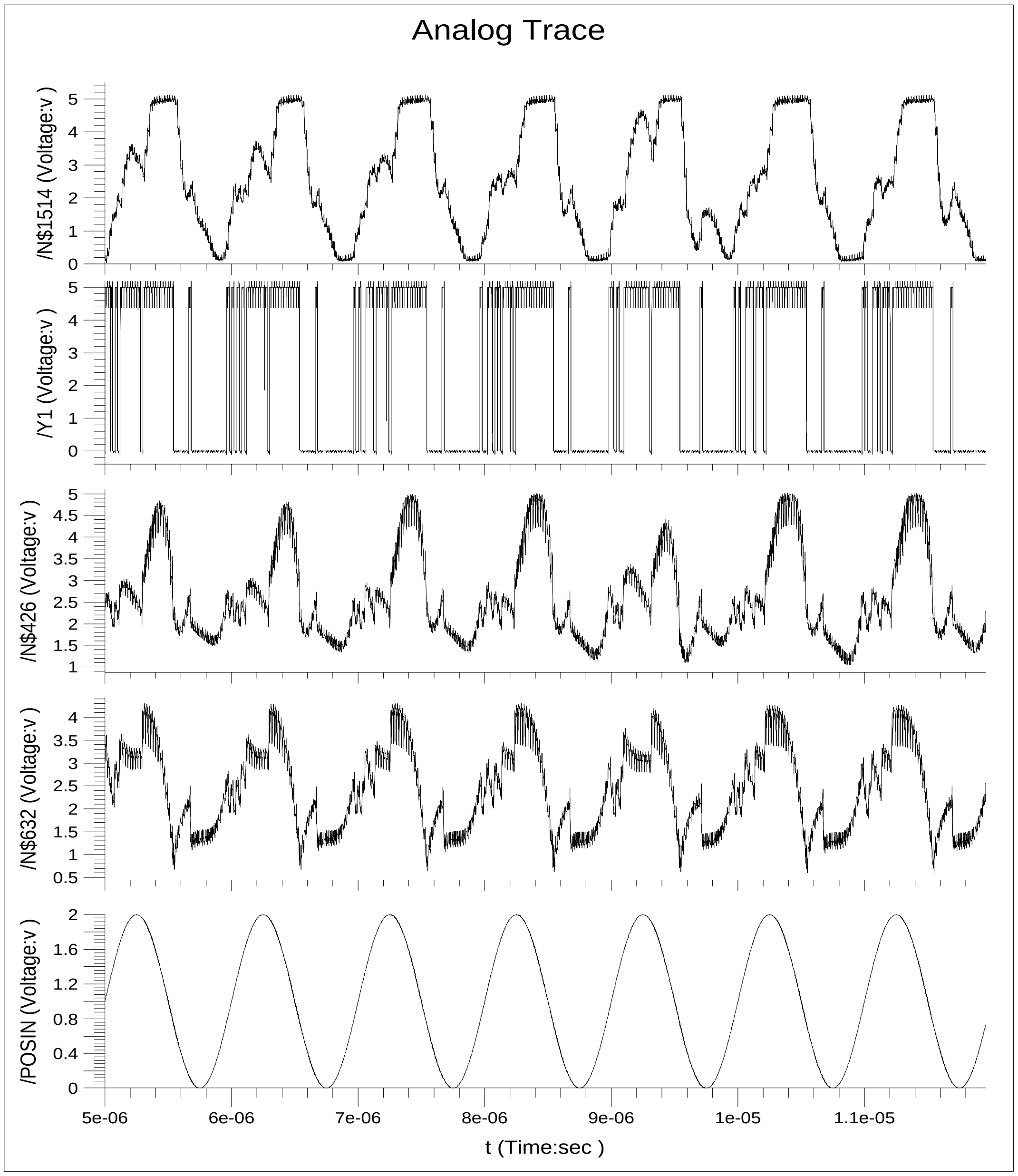


Figure 3.21: Simulation results of the modulator shown in Figure 3.5. (a) Outputs of the integrators and 1-bit quantizer.

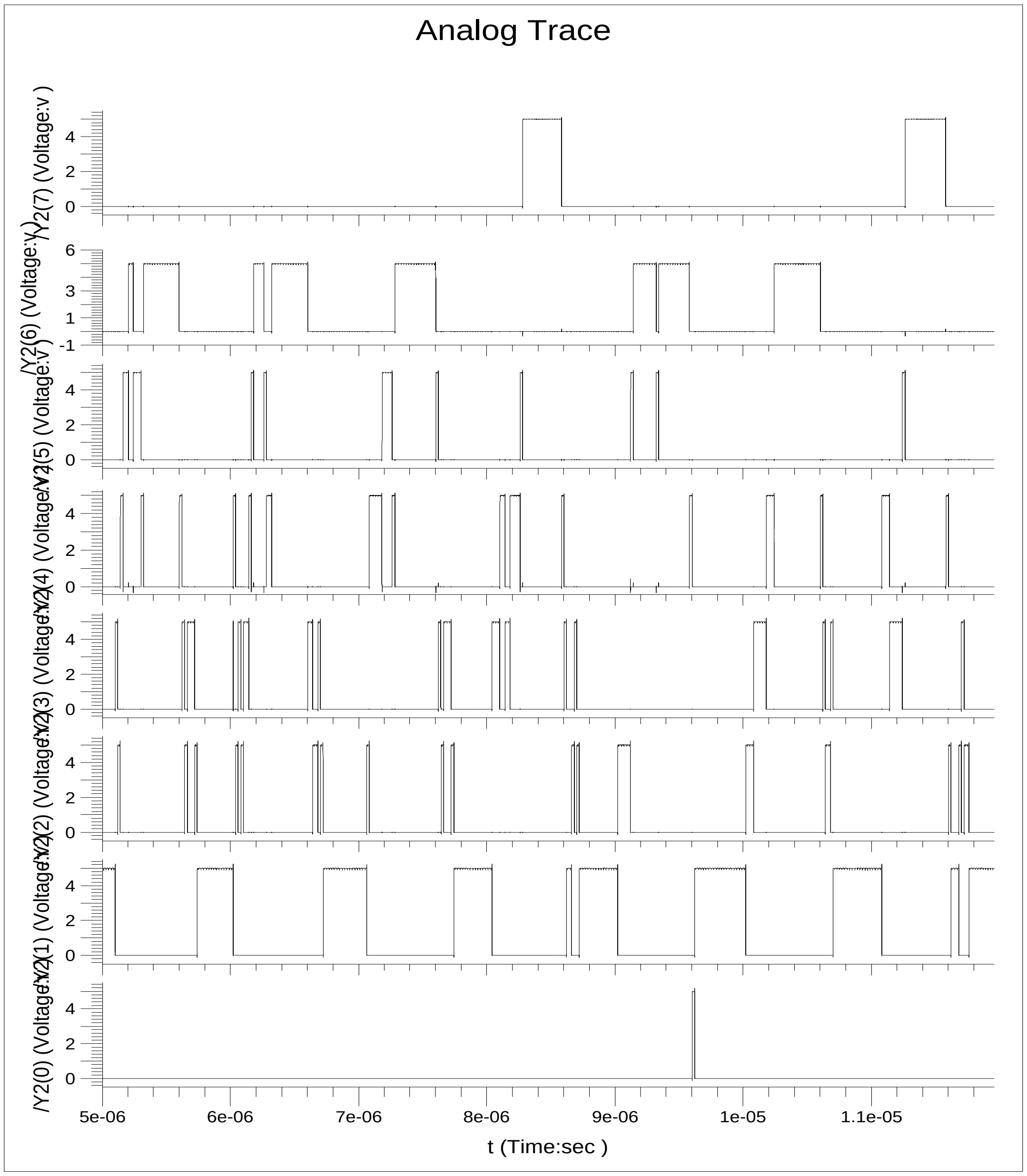


Figure 3.22: Simulation results of the modulator shown in Figure 3.5. (b) Outputs of the 3-bit quantizer.

is combined with it to supply bias voltages. The differential sinusoidal input signals are set at frequency of 1 MHz and magnitude of 2 volts with common-mode voltage CI $= 1V$. The two reference voltages, V+ and V-, are set to $2V$ and $0V$, respectively. As mentioned before, the common-mode output voltage VO is $2.5V$. In Figure 3.21, the positive (upper) signals of the fully differential structure are shown to illustrate the simulation results. From the bottom, they correspond to the input signal, output of the first integrator, output of the second integrator, output of the 1-bit quantizer, and output of the third integrator. We may see that $2V$ (differential) input causes a little bit of overload on the third integrator. If the common-mode input voltage is set to $2.5V$, however, even $1V$ input will cause serious overload on the second integrator. Figure 3.22 shows the 1-out-of-8 code produced by the 3-bit flash ADC.

Chapter 4

DIGITAL DECIMATOR DESIGN

The original implementation of the high-speed $\Delta\Sigma$ ADC done by Brandt was accomplished by processing the digital outputs of the analog modulator on a workstation to perform the error cancellation and decimation filtering. In practical $\Delta\Sigma$ ADC designs, however, digital circuits have to be hardwired to realize the real-time data conversion. In this chapter, the digital error cancellation logic and decimation filters are designed in Matlab, described in VHDL (Very-high-speed-integrated-circuit Hardware Description Language), and synthesized into standard-cell integrated circuits.

4.1 Digital Filter Design

The digital circuit blocks shown in Figure 4.1 are chosen to implement the error cancellation and sampling decimation. In Figure 4.1, the two digital outputs of the analog modulator described in the last chapter are fed into a digital cancellation logic, which is defined by Equation (3.4) and (3.6), or revised as

$$Y(z) = (2z^{-2} - z^{-3})Y_1(z) - 4(1 - z^{-1})^2 Y_2(z) \tag{4.1}$$

where $Y_1(z)$ and $Y_2(z)$ are the 1-bit and 3-bit output in Figure 3.4, respectively.

In the present design, it is desired to decimate the data words of $Y(z)$ from a sampling rate of 50 MHz to one of $50/24 = 2.08$ MHz with a baseband extending from zero to 1 MHz. A linear-phase requirement results in the exclusive use of FIR filters for the decimator. Table 4.1 lists the practical specifications of the decimator. A single-stage implementation of the decimator will result in a filter size of 3050 taps (coefficients) as per Equation (2.21) (see Appendix C.1) with $F_s = 50$ MHz and a

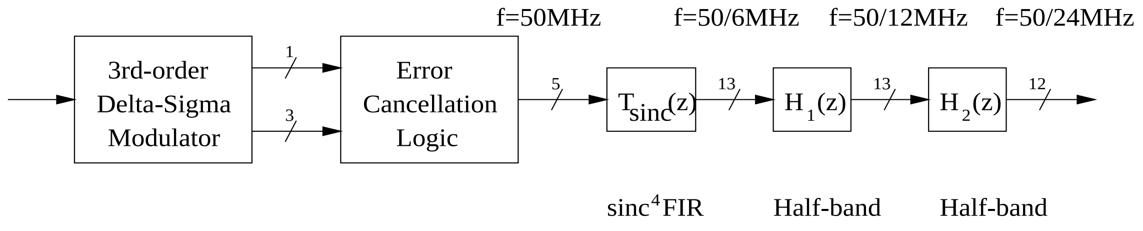


Figure 4.1: Block diagram of digital filters.

computation rate of $3050 \times (50/24) \times 10^6 = 6.355 \times 10^9$ multiplications per second. However, if a three-stage decimator is designed as shown in Figure 4.1, the FIR filters will have fewer orders and less computation. The first 6:1 converter reduces the sampling rate from 50 MHz to 8.33 MHz with a transition region of 1-7.29 MHz as per Equation (2.20). Thus, the first filter requires a filter size of 20 taps and a computation rate of 1.67×10^8 multiplications per second. Equivalently, the two 2:1 converters that reduce the sampling rate from 8.33 MHz to 4.17 MHz and further to 2.08 MHz with transition bands of 1-3.13 MHz and 1-1.04 MHz, respectively, have the respective filter orders of 8 and 255 taps, and computation rates of 3.3×10^7 and 5.31×10^8 multiplications per second. Therefore, the three-stage decimator has a combined filter order of 283 taps and a total computation rate of 7.31×10^8 multiplications per second, i.e., about 11 times and 9 times smaller than those of the single-stage decimator, respectively.

To further reduce the computation rate of the three-stage decimator, special

Table 4.1: Design Specifications of the Decimator.

passband cutoff frequency	F_{pc}	1 MHz
stopband cutoff frequency	F_{sc}	50/48=1.04 MHz
passband ripples	δ_p	0.01
stopband ripples	δ_s	0.001

types of FIR filters, such as comb filter and half-band filter, may be applied to implement the three filters, as we mentioned in Section 2.4.3.

4.1.1 Fourth-Order Comb ($sinc^4$) Filter

As Candy [30] showed, *comb* filters with response $sinc^k(\omega)$ are appropriate for decimating $\Delta\Sigma$ modulation down to four times the Nyquist rate, and the order of the comb filter should be at least one larger than that of the noise-shaping modulator in order to prevent excessive aliasing of out-of-band noise from entering the baseband. Therefore, in the present design, where the oversampling ratio is 24 and the quantization noise is third-order shaped, the first-stage comb filter shown in Figure 4.1 should be fourth-order with a decimation factor of 6 or equivalently, as per Equation (2.23), have a transfer function

$$H(z) = (\frac{1}{6} \times \frac{1 - z^{-6}}{1 - z^{-1}})^4 \tag{4.2}$$

Simulations in Matlab [31] (see Appendix C.2) give the frequency response of this comb filter in Figure 4.2. The upper Bode diagram in Figure 4.2 shows three notches in the region of 0–π (i.e., 0–25 MHz) which are wide and deep enough to cover the stopband width of 2.08 MHz required by the decimator. Whereas, the lower Bode diagram enlarges the frequency response in the region of 0–$\pi/12$ where the in-band signal loss is less than 1 dB so that a compensation filter is not necessary.

The computation rate of this $sinc^4$ filter depends on what kind of structure is adopted to implement it. If a cascaded structure described in Section 4.2 is used, no multiplications are demanded at all such that the filter has a computation rate of 2.33×10^8 additions/subtractions per second.

4.1.2 Half-band Filters

The purpose of the filters following the first-stage comb filter is to remove any higher-frequency input signal (in effect, a sharp anti-aliasing filter) before the signal is

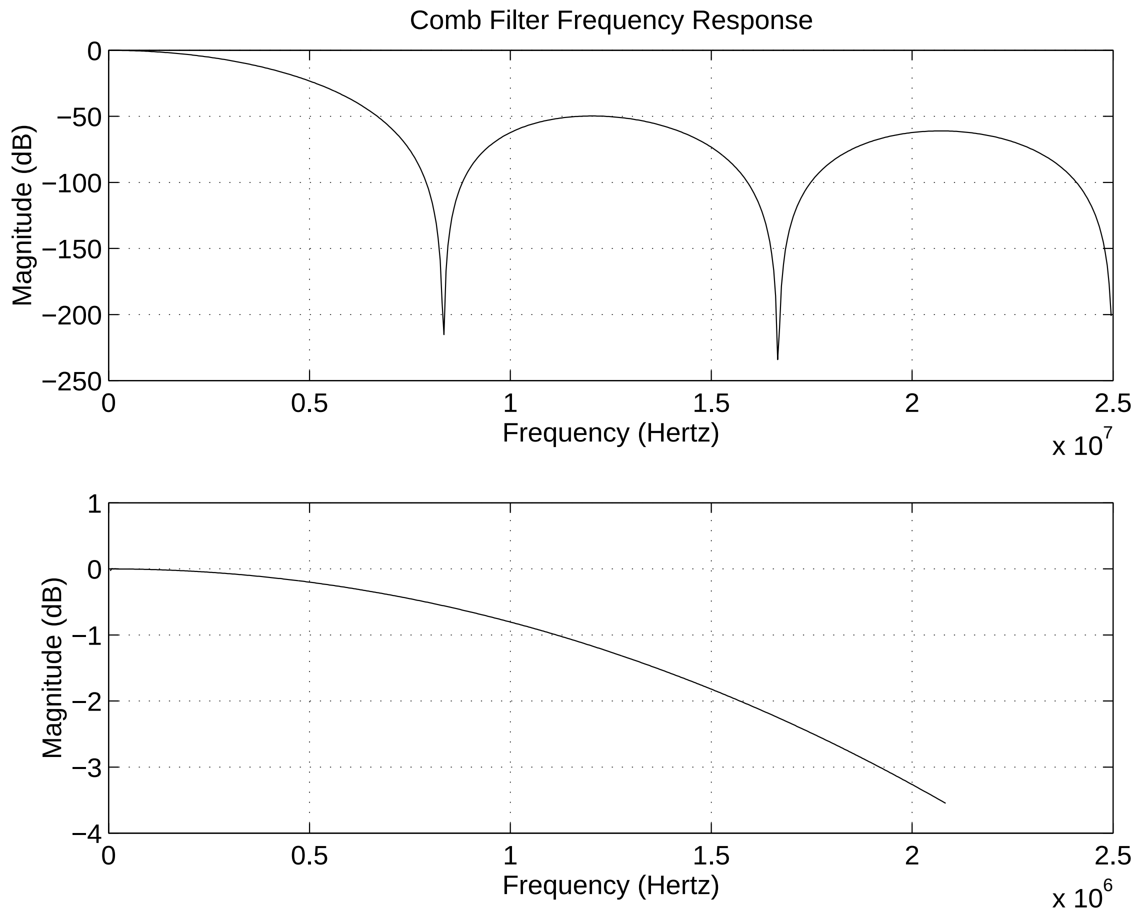


Figure 4.2: Frequency response of $sinc^4$ filter with a decimation ratio of 6.

downsampled to the final Nyquist rate. In other words, while the comb filter is good at filtering out the quantization noise, it is not sharp enough to act as a reasonable anti-aliasing filter for input signals slightly higher than baseband frequency. Instead using arbitrary FIR filters to realize this anti-aliasing filter, a few half-band FIR filters might be used in order to save about half multiplications.

As mentioned in Section 2.4.3, half-band filters have equal passband and stopband ripples as well as equal passband and stopband width. Such filters naturally have the property that approximately half of the filter coefficients are exactly zero. Hence

Table 4.2: Nonzero Coefficients of the 11-tap Half-Band Filter.

Ideal Coefficients	Rounded Coefficients	Binary Coefficients
b(1) = b(11) = 0.012218	0.012207	0.00000011001
b(3) = b(9) = -0.062581	-0.062500	-0.00010000000
b(5) = b(7) = 0.300962	0.300781	0.01001101000
b(6) = 0.500014	0.500000	0.10000000000

* Other coefficients $b(2) = b(4) = b(10) = b(8) = 0$.

the number of multiplications in implementing such filters is half of that needed for a linear phase design. The half-band filter is appropriate only for sampling rate changes of 2 to 1. Thus, the final two filters in Figure 4.1 are preferable to be implemented as half-band filters.

For a two-stage half-band implementation, the same passband and stopband ripples are required as

$$\delta = \min(\frac{\delta_p}{2}, \delta_s) = 0.001 \tag{4.3}$$

where δ_p and δ_s are listed in Table 4.1. For the two half-band filters, the stopband cutoff frequencies are $50/12 - 1 = 3.17$ MHz and $50/24 - 1 = 1.08$ MHz, respectively. Note that the final stopband cutoff frequency is larger than F_s defined in Table 4.1. In other words, the final half-band filter allows aliasing into the transition region F_p to F_s. The respective filter sizes are approximately (based on the next largest odd number as per Equation (2.21)) 11 taps and 165 taps. Therefore, the computation rates for both filters are about 2.3×10^7 and 1.72×10^8 multiplications per second, respectively. If four additions or subtractions are averagely equal to one multiplication, the total computation rate of the comb and half-band filters is about 2.53×10^8 multiplications per second, i.e., approximate 3 times smaller than that of conventional FIR filters.

Table 4.3: Nonzero Coefficients of the 165-tap Half-Band Filter.

Ideal Coefficients	Rounded Coefficients	Binary Coefficients
b(2) = b(164) = 0.000641	0.000488	0.000000000010
b(4) = b(162) = -0.000301	-0.000244	-0.000000000001
b(6) = b(160) = 0.000371	0.000244	0.000000000001
b(8) = b(158) = -0.000450	-0.000488	-0.000000000010
b(10) = b(156) = 0.000540	0.000488	0.000000000010
b(12) = b(154) = -0.000641	-0.000488	-0.000000000010
b(14) = b(152) = 0.000755	0.000733	0.000000000011
b(16) = b(150) = -0.000883	-0.000977	-0.000000000100
b(18) = b(148) = 0.001025	0.000977	0.000000000100
b(20) = b(146) = -0.001183	-0.000977	-0.000000000100
b(22) = b(144) = 0.001357	0.001953	0.000000001000
b(24) = b(142) = -0.001550	-0.001953	-0.000000001000
b(26) = b(140) = 0.001763	0.001953	0.000000001000
b(28) = b(138) = -0.001997	-0.001953	-0.000000001000
b(30) = b(136) = 0.002255	0.002197	0.000000001001
b(32) = b(134) = -0.002537	-0.002441	-0.000000001010
b(34) = b(132) = 0.002847	0.002930	0.000000001100
b(36) = b(130) = -0.003188	-0.002930	-0.000000001100
b(38) = b(128) = 0.003561	0.003906	0.000000010000
b(40) = b(126) = -0.003970	-0.003906	-0.000000010000
b(42) = b(124) = 0.004421	0.004395	0.000000010010
b(44) = b(122) = -0.004917	-0.004883	-0.000000010100
b(46) = b(120) = 0.005464	0.005859	0.000000011000
b(48) = b(118) = -0.006070	-0.005859	-0.000000011000
b(50) = b(116) = 0.006744	0.007812	0.000000100000
b(52) = b(114) = -0.007497	-0.007812	-0.000000100000
b(54) = b(112) = 0.008343	0.008301	0.000000100010
b(56) = b(110) = -0.009302	-0.009766	-0.000000101000
b(58) = b(108) = 0.010398	0.010742	0.000000101100
b(60) = b(106) = -0.011665	-0.011719	-0.000000110000
b(62) = b(104) = 0.013150	0.013672	0.000000111000
b(64) = b(102) = -0.014917	-0.015625	-0.000001000000
b(66) = b(100) = 0.017066	0.017578	0.000001001000
b(68) = b(98) = -0.019746	-0.019775	-0.000001010001
b(70) = b(96) = 0.023198	0.023438	0.000001100000
b(72) = b(94) = -0.027841	-0.027832	-0.000001110010
b(74) = b(92) = 0.034466	0.035156	0.000010010000
b(76) = b(90) = -0.044769	-0.046875	-0.000011000000
b(78) = b(88) = 0.063157	0.062500	0.000100000000
b(80) = b(86) = -0.105800	-0.105469	-0.000110110000
b(82) = b(84) = 0.318209	0.318359	0.010100011000
b(83) = 0.500008	0.500000	0.100000000000

* Other coefficients $b(1) = b(3) = \ldots = b(165) = 0$.

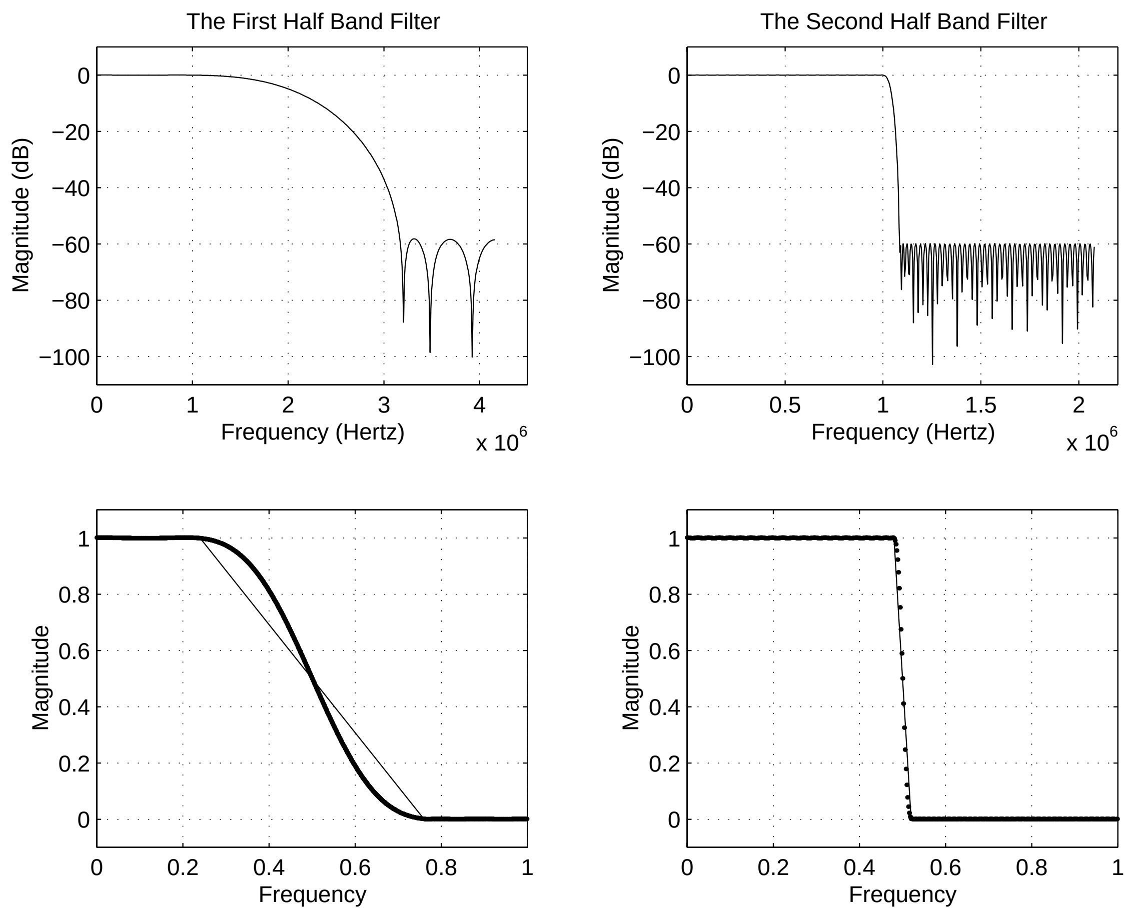


Figure 4.3: Frequency responses of 11-tap and 165-tap half-band filters.

An n-tap digital FIR filter may be expressed as

$$H(z) = \sum_{k=1}^{n} b(k) z^{-(k-1)} \tag{4.4}$$

where the n coefficients $b(k)$ are designed to meet given performance specifications on the filter. There are a few algorithms [20] for determining suitable values of $b(k)$ for linear-phase FIR filters. In this thesis work, the two half-band filters are designed by using the Parks-McClellan algorithm which applies the Remez exchange algorithm and Chebyshev approximation theory to find optimal equiripple filters. The

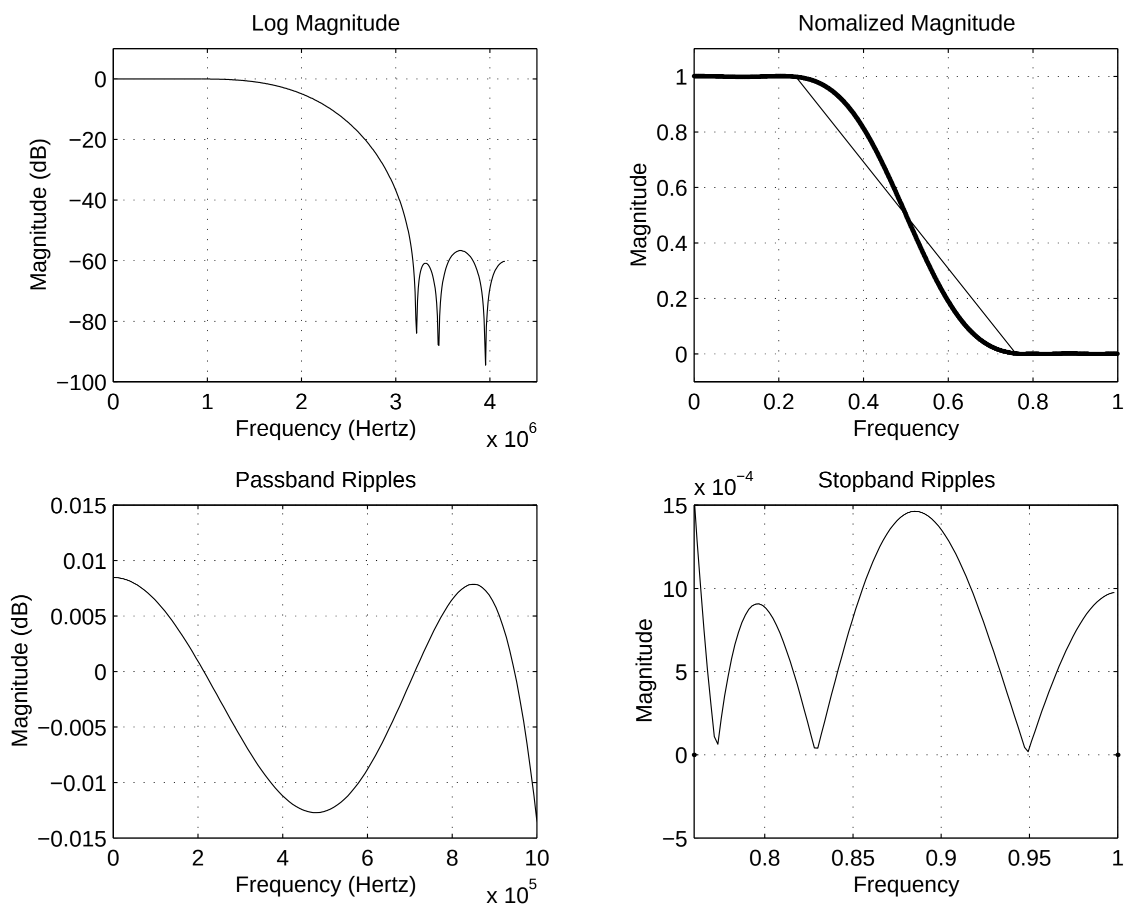


Figure 4.4: Frequency responses of 11-tap half-band filter with rounded coefficients.

11-tap and 165-tap half-band filters are simulated in Matlab (see Appendix C.3) and Figure 4.3 shows the ideal frequency responses of the designed filters. The upper diagrams in Figure 4.3 depict the log magnitude responses in dB, while the lower ones show the normalized magnitude responses versus normalized frequencies (i.e., 1 corresponds to π.) for both desired filters (dashed curves) and designed filters (dotted curves).

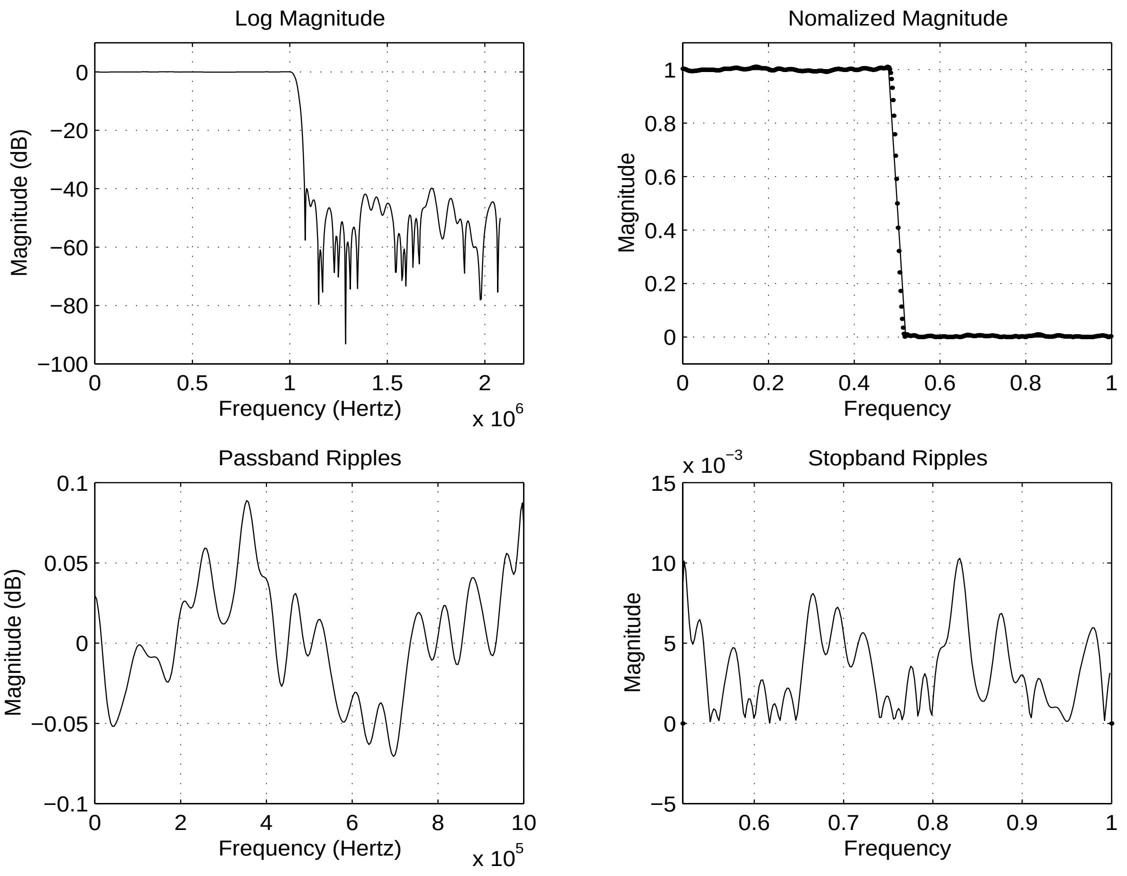


Figure 4.5: Frequency responses of 165-tap half-band filter with rounded coefficients.

The ideal nonzero coefficients of the 11-tap filter are listed in right column of Table 4.2, while those of the 165-tap filter are listed in Table 4.3. The coefficients are symmetrically distributed around $b(\frac{n+1}{2})$ and almost half of them are zero. Since the coefficients have to be represented in binary in order to be implemented into digital hardware, rounded coefficients and their binary correspondences are also listed in Table 4.2 and 4.3 for comparison. The word-lengths of the fixed-point coefficients are set to 11 bits and 12 bits for the 11-tap and 165-tap filters, respectively, due to the overall 12-bit resolution of the system, though optimal selection of the word-length [32, Chapter 11] is not such a simple problem. The rounded coefficients were obtained through empirical iterative simulations (see Appendix C.4) by using as few 1's in each coefficient as possible in order to reduce the computation amount, while maintaining tolerable deviation from the ideal frequency responses. Figure 4.4 and 4.5 depict the resulting frequency responses of the 11-tap and 165-tap filters, respectively.

The lower diagrams in Figure 4.4 and 4.5 show the passband and stopband ripples which are typically smaller than ± 0.01. Simulations reveal that the closer the coefficients are to 0.5 (i.e., $b(6)$ and $b(83)$ in Table 4.2 and 4.3, respectively), the more effect their accuracy have on the frequency responses. Hence those coefficients around that of 0.5 should be rounded as precisely as possible. Sometimes, however, a more accurate value of a specific pair of coefficients may have negative effects on the frequency response. For instance, $b(78)$ and $b(88)$ in Table 4.3 can be more accurate by converting the last second bit to 1, whereas the frequency response at the stopband will drop about 10 dB. Also, a more precise value may have only negligible effects while needing much more 1's in binary number which correspond to more shifts and additions in hardware. For example, $b(64)$ and $b(102)$ should be -0.0000001111101 instead of -0.000001 as shown in Table 4.2, but the additional hardware size introduced by the five more 1's will not be worth of the little resulting improvement on the frequency response. In Table 4.3, many coefficients far away from $b(83)$ are just coarsely rounded without seriously impairing the performance of the 165-tap filter.

4.2 Digital Filter Structures

To realize the transfer functions described above into hardware, proper structures and word-lengths for computing have to be first determined, depending on the type of machine arithmetic used. In this digital implementation, fixed-point two's complement arithmetic is used in order to simplify the hardware design as well as, most importantly, avoid overflow.

Table 4.4: Decimal Equivalents of Numbers 0.11 to 1.00.

Binary	Decimal equivalent (fourths)		
number	Signed-magnitude	1's complement	2's complement
0.11	3	3	3
0.10	2	2	2
0.01	1	1	1
0.00	0	0	0
1.11	-3	-0	-1
1.10	-2	-1	-2
1.01	-1	-2	-3
1.00	-0	-3	-4

4.2.1 Two's Complement Arithmetic

In fixed-point arithmetic, the numbers are usually assumed to be proper fractions. A binary proper fractional number is obtained by setting the binary point between the first and second bits. The first (i.e., the most significant) bit is reserved for the sign of the number. The positive sign is 0, while the negative sign is 1. For example, a positive binary number $+0.010$ is simply represented as 0.010. However, a negative binary number -0.010 may have three forms: 1.010, 1.101, and 1.110 depending on whether the *signed-magnitude*, or *one's complement*, or *two's complement* arithmetic is adopted. Table 4.4 lists the possible 3-bit binary numbers and their decimal equivalents in three arithmetic systems. The signed-magnitude and the 1's complement systems have two representations for zero whereas the 2's complement system has only one. On the other hand, -1 is represented in the 2's complement system but not in the other two.

The two's complement arithmetic can be realized in the simplest hardware compared with the other two, because the carry-out at the sign bit can be ignored in 2's complement addition. This *wrap-around* property is essential to the proper operation of the comb filter [33], as we will mention later. Therefore, the 2's complement arithmetic is used in the whole digital system design.

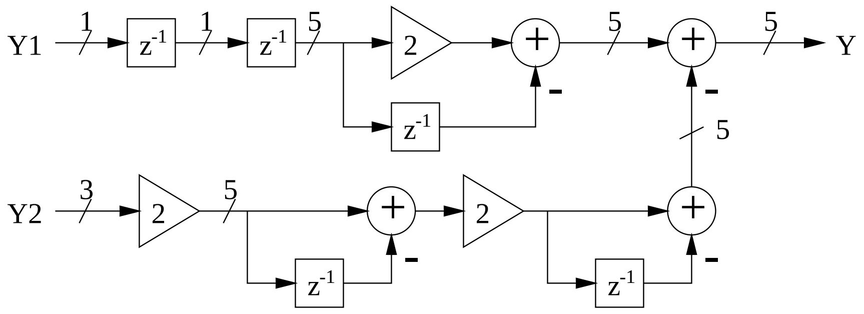


Figure 4.6: Structure for error cancellation logic.

Before choosing structures and wordlength for the decimation filters, we have to determine the length of registers in the error cancellation logic shown in Figure 4.1 and defined by Equation (4.1). Since in the analog modulator the 3-bit Y_2 is actually scaled to one-quarter (i.e., right shift of two bits) of the full signal range of the 1-bit Y_1, 5-bit binary numbers should be used to represent Y_1. Thus, the two levels of Y_1, 1 and 0, correspond to 0.1111 and 1.0000 in binary (or, 31/32 and -1 in decimal), respectively, while the eight levels of Y_2 correspond to the binary numbers in the range of 0.0011–1.1100 (or, from 3/32 to $-1/8$ in decimal) by adding two more bits at left identical with the leftmost bit of the 3-bit word of Y_2. For example, 010 of Y_2 will be transferred to 0.0010, and 101 will be 1.1101. Figure 4.6 shows a simple and direct structure for realizing the error cancellation logic. Note that the $4(1-z^{-1})^2$ term in Equation (4.1) is accomplished by two cascaded integrators with transfer function of $2(1-z^{-1})$.

4.2.2 Pipelining Structure for Comb Filter

A way to realize the comb filter is to re-write Equation (4.2 as

$$H(z) = (\frac{1}{1-z^{-1}})^4 \, (1-z^{-6})^4 \, \frac{1}{6^4} \tag{4.5}$$

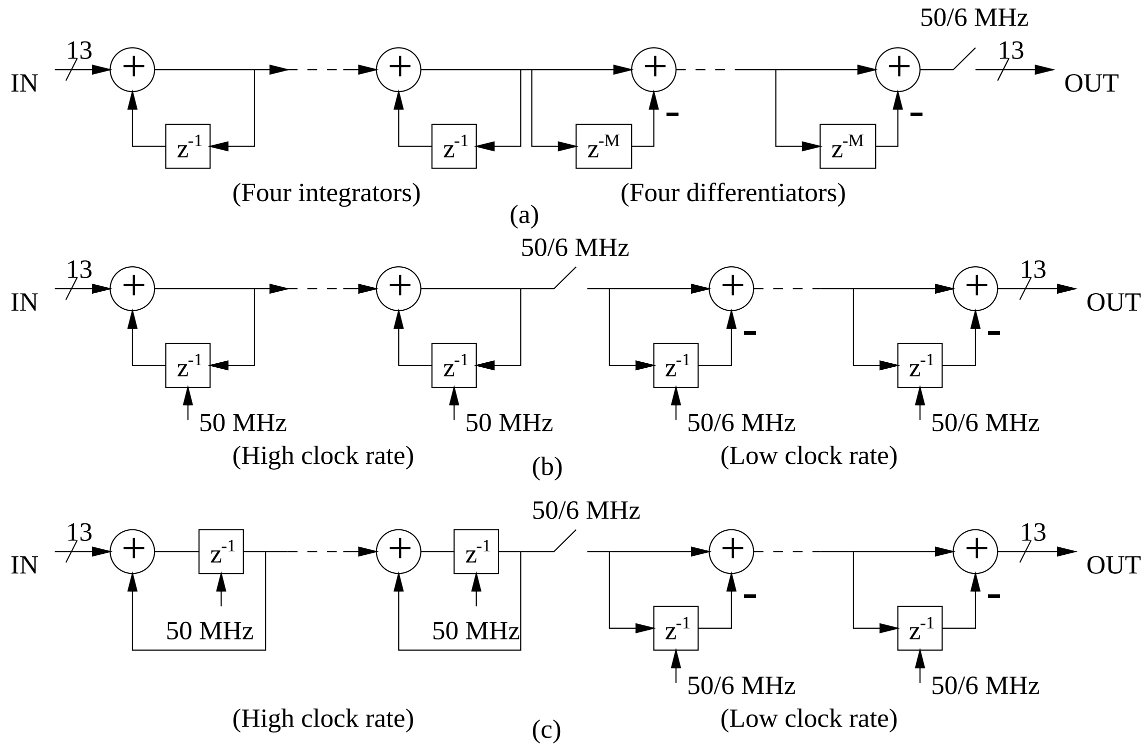


Figure 4.7: Structures for the fourth-order comb filter.

and thus realize it by cascading four integrators (IIR) followed by four cascaded differentiators (FIR) and a dumper as shown in Figure 4.7(a). However, the disadvantage of this architecture is that one should implement $6 * 4$ shift registers for the differentiators. An attractive alternative is depicted in Figure 4.7(b) where the dumper (a switch working at lower frequency) is moved to between the IIR and FIR parts and therefore z^{-6} can be replaced by z^{-1}, resulting in less memory. Since the integrators are working at 50 MHz in this design, the performance requirements for the non-delaying integrators (i.e., four additions per cycle) place heavy burden on their design. Instead, they can be replaced by four delaying integrators [34], as shown in Figure 4.7(c), which will simply introduce four clock delays while allowing one addition per cycle. This pipelining structure for comb filters is more attractive for

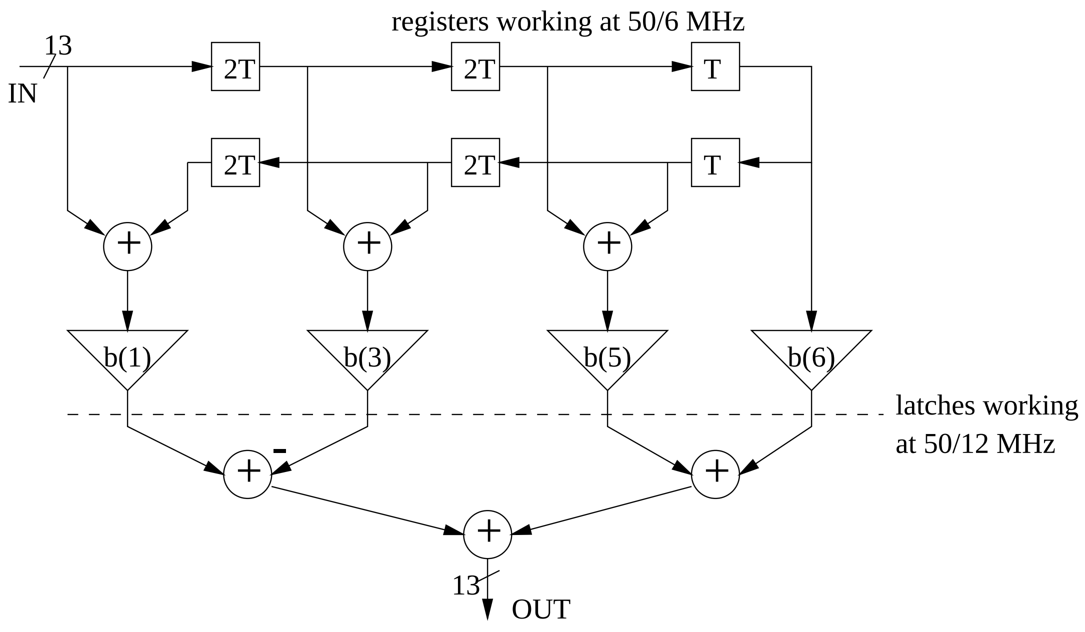


Figure 4.8: Structure for the 11-tap half-band filter.

applications with higher speed.

A point to note here is that at first glance it appears as though the recursive integrators may overflow due to a dc input. Fortunately, it can be proven [33] that overflow can be avoided by using a modulo arithmetic system with wrap-around characteristic everywhere in the system. A sufficient condition for the filter to work correctly is to choose a modulo which is larger than $(M^K + 1) * N$, where M is the decimation factor, K is the order of the comb filter, and N is the dynamic range of the input signal. In the present design, by taking a modulo $2^{13} \geq (6^4 + 1) * 5$, all calculations can be performed by ordinary 2's complement operators without carry handling. Thus, the wordlength for the decimator system is set to 13-bit.

4.2.3 Parallel and Pipelining Structure for Half-Band Filters

Since half-band filters are symmetrical FIR filters, the efficient structure shown in Figure 2.14 can be used to realize the 11-tap and 165-tap half-band filters. Fig-

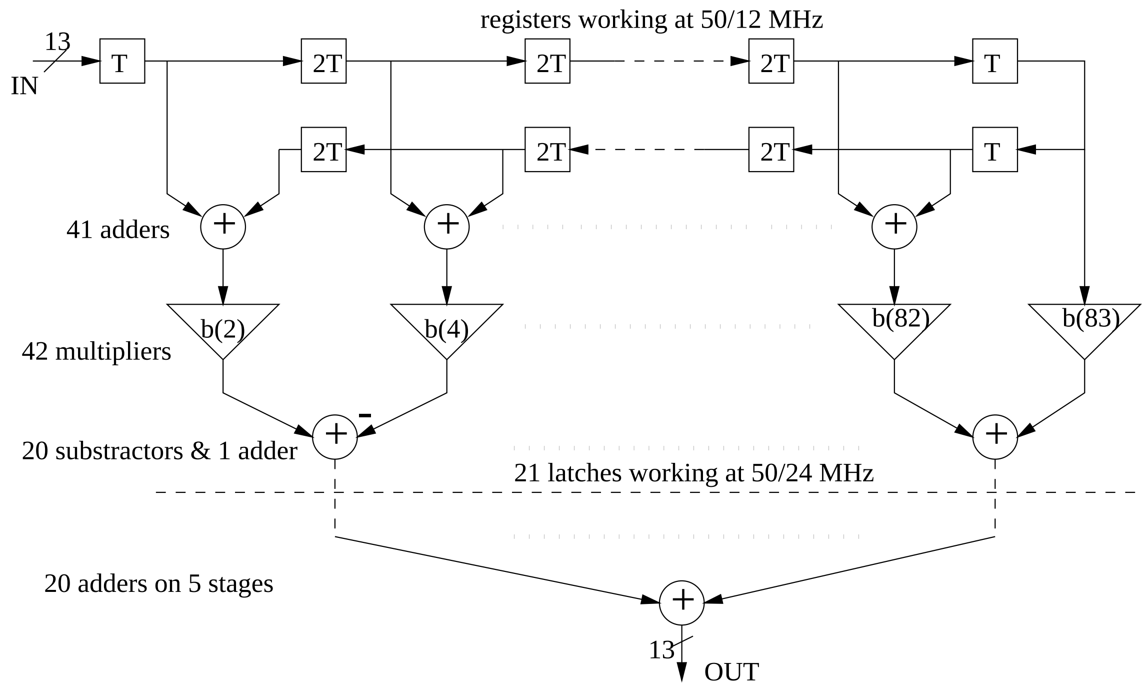

registers working at 50/12 MHz
13
IN
T
2T
41 adders
42 multipliers
b(2)
b(4)
b(82)
b(83)
20 substractors & 1 adder
21 latches working at 50/24 MHz
20 adders on 5 stages
13
OUT

Figure 4.9: Structure for the 165-tap half-band filter.

ure 4.8 indicates the structure of the 11-tap filter with absolute coefficients shown in it. A two-period delay is used between the nonzero coefficients because every second coefficient is zero, except for the middle coefficient $b(6)$, as listed in Table 4.2. The negative coefficient $b(3)$ is realized by performing a positive multiplication followed by a subtraction. All of the additions denoted by symbol $\oplus$ are performed in parallel, i.e., every $\oplus$ represents a 13-bit adder or subtractor. The registers denoted by T are working at 50/6 MHz, while the calculations may work at 50/12 MHz with a decimation factor of 2. To alleviate the performance requirements, a pipelining configuration is also used by inserting a layer of latches to separate the calculations into two stages as shown in Figure 4.8. The latches are working at 50/12 MHz.

All of the multiplications are also performed in parallel via many shifters and adders to avoid the use of a shared multiplier. This parallel and pipelining structure is not area-efficient, but may maximize the speed of calculations so as to be critical for modern high-speed DSP applications.

A similar structure for the 165-tap filter is shown in Figure 4.9. In this structure, more parallel adders and pipelining latches are needed, and the parallel multiplications will consume a large portion of silicon area. Hence the binary coefficients should be rounded with as few 1's as possible because each 1 means a 13-bit adder. Due to the overall 12-bit resolution, the final output of this filter is truncated to 12 bits by ignoring the rightmost bit of the 13-bit word.

4.3 VHDL Implementation

The functionalities of the digital filters are first described on RTL (Register Transfer Level) level via VHDL, and finally implemented into standard-cell integrated circuits. The VHDL programs (see Appendix B.2 to B.5) are coded, compiled, and simulated within Mentor Graphics **QuickHDL**, a VHDL compiler and simulator. The compiled codes are then fed to a synthesizer **AutoLogic II** to get the actual hardware from a MOSIS 0.8-μm standard cell library.

4.3.1 Programming and Simulation

VHDL is an acronym for *VHSIC Hardware Description Language* and VHSIC stands for *Very High Speed Integrated Circuit*. This language was originally developed as a result of a DOD (the Department of Defense) project and now is an IEEE standard (IEEE Std 1076-1993) for describing digital designs. In this section, only some elements of the language that are used in my programs will be explained. Detailed introduction of VHDL may be found elsewhere [35]. Since the synthesizer **AutoLogic II** can only support a subset of the standard VHDL syntax [36], many features of the standard language are not applicable so that the codes are not so compact as they should be.

Error Cancellation Logic

Appendix B.2 gives the VHDL codes for the error cancellation logic defined by Equation (4.1). The **library** command makes two design libraries, *ieee* and *arithmetic*, visible in this design unit. The *ieee* library contains the package *std_logic_1164*, which is an IEEE standard (IEEE Std 1164-1993) and defines a nine-value logic type and its associated overloaded functions and other utilities. Although the *ieee* design library is not part of the VHDL language standard, it is extensively used in digital designs. The *arithmetic* design library and its *std_logic_arith* package, which are developed by Mentor Graphics and associated with Mentor tools, contain synthesizable functions for 2's complement arithmetic and other types of arithmetic operations. This library makes the coding more concise in the present calculation-oriented application. The use commands following the **library** command make visible all items declared in the two packages, *std_logic_1164* and *std_logic_arith*.

After library and package declarations, an **entity** named *e_cancel* is declared. The **entity** declaration describes the external view of the entity, for example, the input signals $y1$, $y2$, and *clk* in *e_cancel*. A **generic** in the entity declaration declares a constant object *multi_bit* with default value of 4, which may be overridden, if different wordlength is needed, while maintaining other codes unchanged.

An entity is modeled using an entity declaration and at least one **architecture** body. The architecture body contains the internal description of the entity, as either a set of interconnected components that represents the structure of the entity, or a set of concurrent or sequential statements that represents the behavior of the entity. In the architecture *behav* of entity *e_cancel*, a behavioral model is described via a combination of concurrent and sequential statements.

At the beginning of the architecture, some internal signals between the primary inputs and outputs are declared. The signals $y1\_d0$ and $y2\_d0$ are used to latch the inputs $y1$ and $y2$ because the analog modulator does not have registers at its output interface. The signals $y1\_d1$ to $y1\_d3$ are outputs of the three delay elements for $y1$ as shown in Figure 4.6. The signal $y2\_d1$ is the output of the delay element in the first differentiator, while the signal $sum1_d$ is that in the second differentiator.

The first three concurrent statements in the architecture define all of the calculations involved in this logic. The signal $sum1$ is the output of the first differentiator, while the signal $sum2$ combines all other summation outputs. Note that the gain factor of 2 is accomplished by a left shift of 1 bit via the **sla** operator (shift left arithmetic).

The **process** statement contains sequential statements that describe the functionality of a portion of the entity in sequential terms. This process is triggered by a rising edge on the clock signal clk. In this process, the 1-bit input $y1$ is extended to 5-bit $y1\_d2$ by converting 1 to 01111 and 0 to 10000, while the 3-bit input $y2$ is extend to 5-bit $y2\_d0$ with a gain of 2. The primary output y is a latched version of $sum2$.

Comb Filter

The VHDL program for the fourth-order comb filter is given in Appendix B.3. Actually, this program can be a general model for any comb filter by applying different values to the generic objects: the input wordlength d, the order of comb filter k, and the output wordlength l. Two clock signals, $clk1$ and $clk2$, are used in this filter at the original sampling rate and the decimated sampling rate, respectively.

A new data type named *sinc_adder* is defined as an unconstrained array of another constrained array of *signed*. Four internal signals are declared as this kind of array of array. The signal $sum\_d(0)$ is actually a 13-bit version of the 5-bit input x, while other four elements of the sum_d array are outputs of the delay elements in the integrators as shown in Figure 4.7(c). The signal $diff(0)$ is the output of the dumper, while other $diff$ elements contain results from the summation nodes of the cascaded differentiators. The four *sum* signals are resulted from the four adders of the cascaded integrators. Array $diff\_d$ are used to store the results of $diff(0)$ to $diff(3)$.

All of the eight $\oplus$ nodes in Figure 4.7(c) are generated by the **for** statement labeled as *gk*. The clocked process is triggered by rising edges on both $clk1$ and $clk2$. The memory elements in integrators are working at $clk1$, while those in differentiators are working at $clk2$. Note that a reset signal rst is used by the memory elements in the integrators, because these registers have to be initially set to known states as long as they are configured in feedback paths. Otherwise, the initial unknown states will be iterated and propergated forever. The registers in the differentiators do not need to be initially reset due to their feedforward configurations. Between the two stages of memory elements, $diff(0)$ and y, the maximum combinational delay is four times the delay of a 13-bit adder, which should be less than the $clk2$ period. This constraint may be satisfied in the 0.8-μm technology for the present speed. If higher speed is required, delays should be introduced into the differentiators.

Half-Band Filters

In Appendix B.4, two different architectures, *hbf11* and *hbf165* which correspond to the 11-tap and 165-tap half-band FIR filters, respectively, are sharing a common entity declaration for the entity *fir* since the two filters have similar external interfaces. In the architecture *hbf11*, the register file x_d for storing the input x is working at $clk1$, while the latches $sum\_d(i)$ and output y are working at $clk2$, whose frequency should be half of that of $clk1$ for half-band filters. It should be noted that all the multiplications of the coefficients listed in Table 4.2 are accomplished by shift-and-add operations. Each '1' in a coefficient corresponds a right shift operation via the shift operator **sra** (shift right arithmetic). The maximum combinational delay in the 11-tap filter is three times the delay of a 13-bit adder. Similarly, the maximum combinational delay in the 165-tap filter, which is introduced by the assignment statement for y, is five times the delay of a 13-bit adder. However, these constraints are not the speed bottle-neck for the whole digital circuits since both filters are working at low sampling frequency compared to the error cancellation logic.

Finally, all the above design entities and architectures are compiled and stored in the default working library **work**, and then used as components in the structural model for the whole digital design as given in Appendix B.5. In the architecture *struc* of the entity *filter*, four components labeled as $i1$–$i4$ are directly instantiated with the entity-architecture pairs described above. A clock divider is modeled in the **block** *clk_gen* to generate three decimated clock signals from the original clock *clk*, which may come from the $C2N$ signal in Figure 3.7.

Simulation

To simulate the VHDL models on RTL (Register Transfer Level) level, digital stimulus have to be derived from the outputs of the analog modulator. However, the complicated digital waveforms shown in Figure 3.21 and 3.22 can hardly be modeled into regular and periodical test patterns. Figure 4.10 shows the digital outputs of the modulator as a differential dc signal of $0.666V$ with reference range of 0–$2V$ is applied to the input. A periodical pattern may be observed from the output $Y1$, while another output $Y2$ appears more irregularly. In order to simplify the digital simulation, I pick up the sequences from 1.48 μs to 1.72 μs in Figure 4.10 as periodical stimulus applied to the digital circuit. Table 4.5 lists the test patterns for inputs $x1$ and $x2$ of the *filter* entity. It should be noted that if a high on $x1$ means a decimal number $+1$ and a low means -1, then the average of the test sequence on $x1$ is $+1/3$, corresponding to the analog dc input $0.666/2 \doteq 1/3$.

The RTL simulation is performed in **QuickHDL** by applying the above test patterns along with clock and reset patterns. A portion of the simulation results is

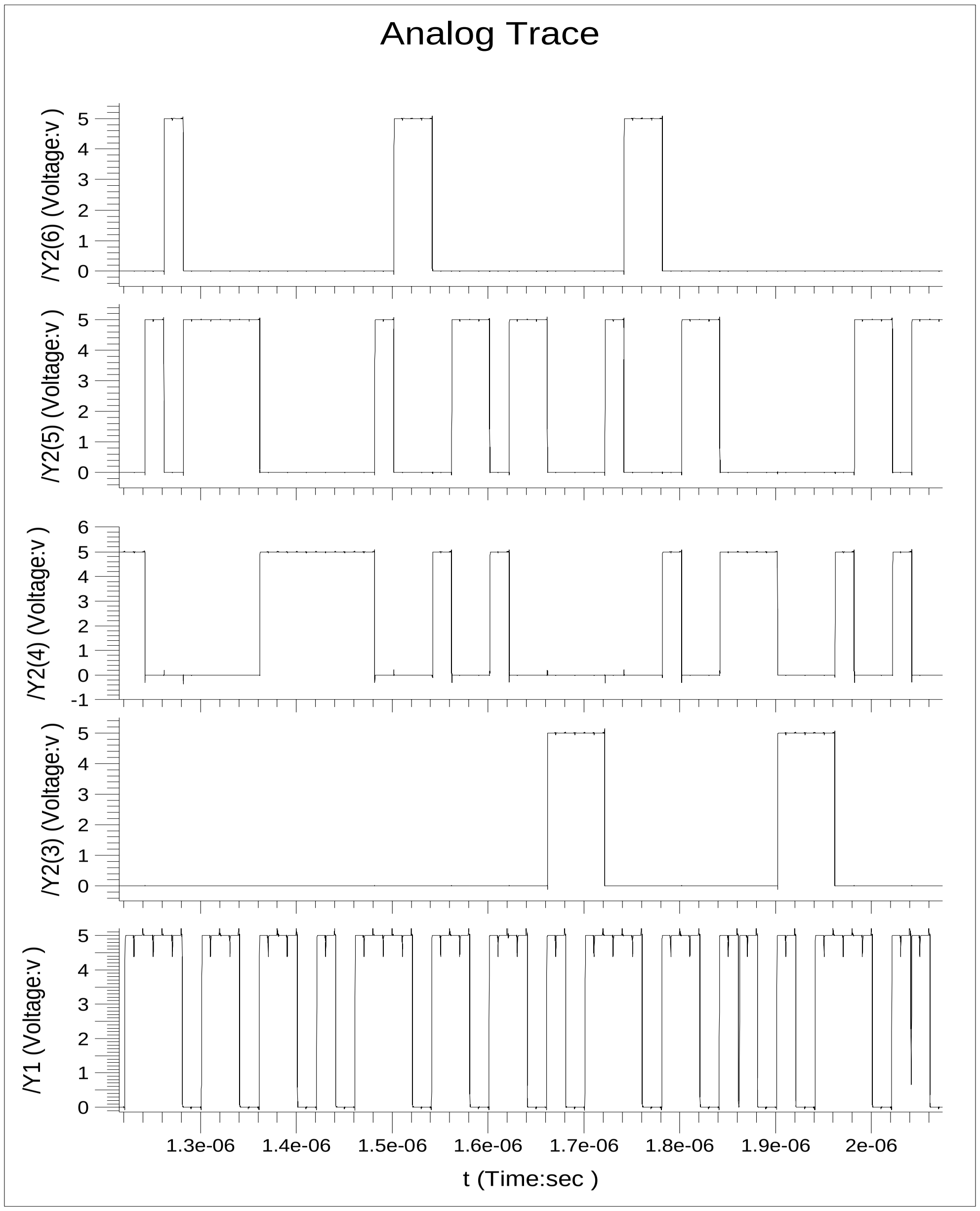


Figure 4.10: Simulation results of the modulator shown in Figure 3.5 with dc input of $\frac{1}{3}$.

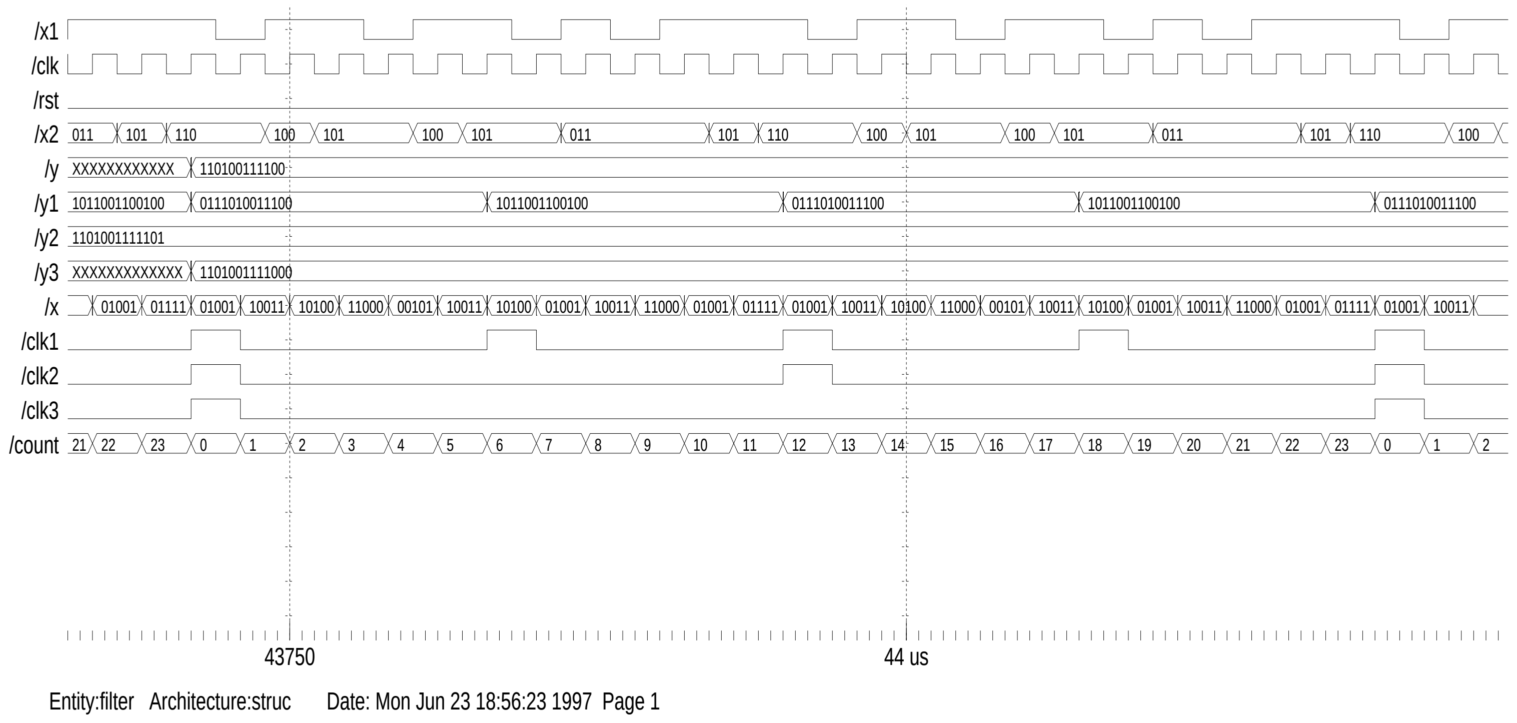


Figure 4.11: Simulation results of the digital circuits modeled in Appendix B.5.

Table 4.5: Periodical Test Patterns for Digital Filters.

cycle	1	2	3	4	5	6	7	8	9	10	11	12
x1	1	1	0	1	1	0	1	1	0	1	0	1
x2	101	110	110	100	101	101	100	101	101	011	011	011

shown in Figure 4.11. The signals $clk1$–$clk3$ are clocks dividing clk by 6, 12, and 24. The signal x is the output of the error cancellation logic, while the signals $y1$–$y3$ are the outputs of the comb, 11-tap, and 165-tap filters, respectively. Note that $y3$ shows up meaningful results after the simulation has run for 43700 ns. This is because all the registers in feedforward configurations are not initially reset to known states. The primary output y is a 12-bit truncated version of $y3$. The output is 1.10100111100 in binary and -0.345 in decimal, approximately an inversion of $1/3$ with offset error. The inversion is probably introduced by the interface between analog modulator and digital filters, while the offset error is due to the inaccurate test patterns in Table 4.5.

4.3.2 Synthesis

Now it is ready to synthesize the compiled VHDL codes into ASIC (Application Specific Integrated Circuits) or FPGA (Field Programmable Gate Array) design. The synthesis process in Mentor Graphics **AutoLogic II** involves generic netlist synthesis, area optimization and timing (performance) optimization [37] based on a destination technology. A cell library is needed to map the synthesized generic structure into a netlist of logic gates. Here we are using the 0.8-μm standard cell library from MOSIS, *scn08hp_lib*. The optimized netlist may be stored into a proper format, such as EDIF, EDDM, and GENIC, so as to facilitate further gate-level simulation and layout design. It takes about 6 hours for a Sun SPARC 20 workstation to finish the synthesis and area optimization (with low effort on logic factoring) for the architecture *struc* of entity *filter*. To illustrate how large the resulting circuit is, Table 4.6 gives

Table 4.6: Global Cell Usage Statistics for Digital Filters.
Cost in Area Units: $10^3 \mu m^2$

Cell Name	Instance Count	Cost/ Cell	Subtotal
GFL_LIB:FALSE	47		
SCN08HP_LIB:and02	81	0.76	61.16
SCN08HP_LIB:ao012	8	0.94	7.54
SCN08HP_LIB:ao013	3	1.13	3.40
SCN08HP_LIB:ao022	145	1.13	164.14
SCN08HP_LIB:ao023	7	1.32	9.25
SCN08HP_LIB:aoi012	465	0.76	351.07
SCN08HP_LIB:aoi022	14	0.94	13.20
SCN08HP_LIB:dff0	2709	2.27	6135.89
SCN08HP_LIB:inv01	3957	0.38	1491.79
SCN08HP_LIB:mux0	6	1.32	7.93
SCN08HP_LIB:nand02	1220	0.57	690.52
SCN08HP_LIB:nand03	1	0.76	0.76
SCN08HP_LIB:nand04	1	0.94	0.94
SCN08HP_LIB:nor02	12	0.57	6.79
SCN08HP_LIB:nor03	1	0.76	0.76
SCN08HP_LIB:oa012	2	0.94	1.89
SCN08HP_LIB:oa013	1	1.13	1.13
SCN08HP_LIB:oa014	1	1.32	1.32
SCN08HP_LIB:oa023	1	1.32	1.32
SCN08HP_LIB:oai012	1756	0.76	1325.78
SCN08HP_LIB:oai013	16	0.94	15.09
SCN08HP_LIB:oai014	1	1.13	1.13
SCN08HP_LIB:oai022	28	0.94	26.40
SCN08HP_LIB:oai023	7	1.13	7.92
SCN08HP_LIB:oai0113	1	1.13	1.13
SCN08HP_LIB:or02	4	0.76	3.02
SCN08HP_LIB:xnor0	146	1.32	192.87
SCN08HP_LIB:xor0	1094	1.32	1445.17
Total:	11735		11969.30

Table 4.7: Hierarchy Statistics for Digital Filters.

Cost in Area Units: $10^3 \mu m^2$

Inst name	Cell name	local prim count	local prim cost	local module count	local module cost	global prim count	global prim cost
/	filter.struc	35	32.08	4	11937.22	11735	11969.30
i1	e_cancel.behav	53	69.64			53	69.64
i2	decimator.opt	325	394.04	8	377.39	847	771.43
i3	fir.hbf11	426	553.49	10	434.26	945	986.75
i4	fir.hbf165	5039	6529.63	88	3579.79	9855	10109.42

the statistical report for the standard cells used in the circuit, while Table 4.7 shows how large each of the four major components is.

From Table 4.6 we may see that the circuit consists of 11.7 K gates consuming about 12 mm^2 silicon area. If the routing area of a standard-cell ASIC design is estimated as 40% of the total silicon area, this digital core circuit will cost totally about 20 mm^2 silicon. It is typical for a transistor in this library to cost about 120 μm^2 area with overhead. Thus, the total number of transistors in this circuit is about 0.1 million. The majority of gates used are D-flipflops (*dff0*), inverters (*inv01*), XOR gates (*xor0*), and OAI gates (*oai012*), which are major building blocks for registers and adders. In Table 4.7, a primitive (*prim*) corresponds to a standard cell (or gate), while a module means an adder or a subtractor consisting of many primitives. For example, the 11-tap half-band filter (*i3*) contains 426 primitives (mainly *dff0* and *inv01*) as well as 10 modules which are constructed by other $945 - 426 = 519$ primitives (mainly *xor0* and *oai012*). It should be noted that the 165-tap FIR filter consumes up about 90% of the total resources, while the comb filter costs less than 7% of the area.

One or two more area optimizations may be performed to achieve less silicon consumption, but the effects will not be very obvious because this design is mainly constructed by registers and adders which are already optimum. Timing optimizations, which are usually performed after area optimizations by using larger area to prevent any violations of timing constraints, may not be needed due to the pipelining structures inside the circuits.

Chapter 5

CONCLUSION

This thesis dealt with the design of a $\Delta\Sigma$ analog-to-digital converter for high-speed applications, using 0.8-μm CMOS VLSI technology from MOSIS. A complete $\Delta\Sigma$ A/D converter combines an analog $\Delta\Sigma$ modulator with a digital decimator to construct a mixed-mode signal processing system. This work accepts the challenge of designing with such a mixed-signal system of analog CMOS circuits and digital signal processing filters.

5.1 Summary

A published $\Delta\Sigma$ modulator structure [1] was implemented into VLSI. This modulator is an interesting structure because it may be applied to high-speed data conversions while most of the commercial $\Delta\Sigma$ data converters were used for low-speed applications like digital telephony and digital audio. The work was to migrate that published modulator into a faster technology with necessary modifications, and complete the $\Delta\Sigma$ ADC system by creating a new DSP architecture for the high-speed application.

Based on the specifications and interface of the published $\Delta\Sigma$ modulator, a new digital decimator was designed and verified first. There are many choices to implement a decimator, and a three-stage linear-phase structure was chosen in this work. The digital filters realizing the decimator include a fourth-order comb filter with decimation factor of 6, an 11-tap half-band FIR filter, and a 165-tap half-band FIR filter. These filters were generated and simulated in Matlab. Then they were modeled in VHDL, verified with an RTL simulator, and synthesized into a standard-cell circuit. Mentor Graphics tools were used in the thorough design procedure.

A common way to implement a digital filter is to store the coefficients in ROM (read-only memory) and perform the multiplications in a dedicated multiplier. In this work, however, a pipelining, parallel, and multiplierless structure was used to implement the half-band filters, since this structure might be more feasible for higher-speed applications when finer VLSI technologies are available. The synthesized digital circuit contains about 0.1 million transistors and consumes about 20 mm^2 silicon area with 0.8-μm standard-cell technology.

The most difficult part of this work was the analog modulator design, not only because analog IC design is a kind of "empirical art", but also because the original paper proposing that high-speed $\Delta\Sigma$ modulator did not give so many details that sometimes there were some kind of uncertainty in the design. For example, as the analog circuit was initially simulated under a differential input of $3V$ with the common-mode voltage set at $0V$, the 3-bit output $Y2$ just bounced between two levels. This problem was solved by re-defining the input range to $0-2V$ while keeping the integrators from overload.

Although most of the original analog circuits were mapped into the 0.8-μm technology without structural modifications, some new features were added. A wide-swing bias circuitry was added so that the original opamp design was modified to fit in with the wide-swing structure. Hence, lots of simulations have been performed to look for the proper bias voltage values for the opamp's operation. A new structure published in the 90's has been accommodated into the original switched-capacitor integrator to accomplish a combined 1-bit DAC for better performance, and therefore the 3-bit DAC was also modified to take advantage of this upgrade. The final analog circuit contains about 0.5 K transistors, capacitors, and resistors. The modulator was analyzed in Mentor's analog simulator **AccuSim II**, though a mixed-mode simulator is strongly needed to perform the simulation for the whole converter.

So far, the front-end design of this $\Delta\Sigma$ converter, i.e., the design entry and pre-layout simulation, has been done. The back-end design, i.e., the layout design and post-layout simulation, is left for future work. As an estimation of the difficulty of full-custom layout design, the opamp block was laid out as shown in Figure 5.1. This layout has passed all DRC (design rules checking) and LVS (layout versus schematic) checkings. Some analog layout design considerations in this layout will be discussed in the next section.

After all, some highlights of this work may be listed as follows:

- A published $\Delta\Sigma$ modulator was migrated to a faster, 0.8-μm CMOS process.
- A three-stage decimator was designed to complete the $\Delta\Sigma$ A/D converter.
- Some new analog circuits, such as the wide-swing constant transconductance bias circuit and the switched-capacitor 1-bit DAC, were added into the original modulator design.
- Parallel and pipelining structures were implemented in VHDL for the filters.
- Both full-custom (bottom-up) and semi-custom (top-down) IC design approaches were utilized in designing this mixed-signal system.

5.2 Future Work

It is well known that designing a layout is a tedious, time-consuming, and error prone task. For a pure digital design, many mature CAD tools are available to facilitate and even automate the design for smaller and/or faster layout. For analog or mixed analog-digital designs, however, considerations for accuracy have to be carefully taken into account in the layout design phase [4, Chapter 11][5, Chapter 2][3, Chapter 11]. Thus, many special techniques for analog layout, such as a interdigitized fashion for differential pairs, a serpentine arrangement for resistors, and

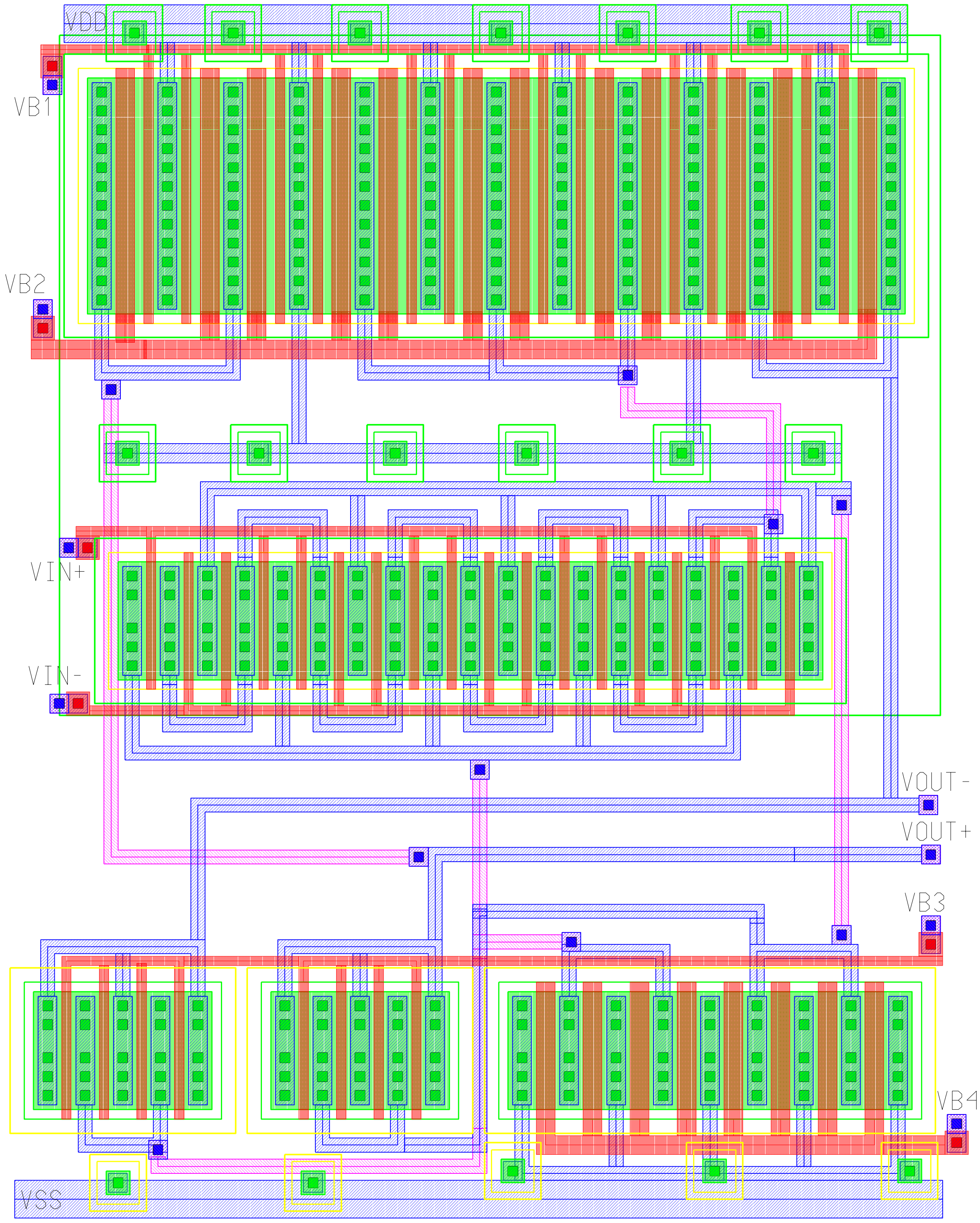


Figure 5.1: Layout for the opamp shown in Figure 3.11.

common centroid structure for capacitors, make only "manual" layout design feasible in some circumstances.

A SDL (schematic-driven layout) procedure in **ICstation**, a Mentor Graphics tool for IC layout design, facilitated the layout process of Figure 5.1, because the device generator available in the current MOSIS design kit can automatically fold large transistors into smaller ones with shared diffusion regions, as shown in Figure 5.1. However, the interconnections of these folded transistors had to be manually routed. For example, the two transistors, $Q1$ and $Q2$ in Figure 3.11, which are made up of a differential pair, were physically implemented into the row of transistors in the middle of Figure 5.1 and connected in an interdigitized fashion to minimize the offset caused by mismatch. The serpentine *metal1* (blue layer) path in the middle row, which has the same stipple pattern as that of signal **VOUT+** and **VOUT-**, and the *poly* (red layer) combs of signal **VIN+** and **VIN-**, had to be carefully drawn "by hand". The total area for this opamp layout is about $85 \times 105 \mu m^2$.

The current 0.8-μm process file is not suitable for analog layout. For example, Appendix A.1 just gives the crossover capacitance between *metal1* and *poly* layers as $0.0104 fF/lambda^2$ (i.e., $0.65 fF/\mu m^2$), while a typical capacitance between *poly* and *electrode* for analog linear capacitors is about $0.5 fF/lambda^2$. Also, the sheet resistance of n-diffusion (0.368 Ω per square) in Appendix A.1 is much smaller than a typical resistance of 2–3 Ω per square. Hence a more accurate and complete process file is needed for analog layout design.

The layout for the analog modulator should be designed in a full-custom manner, while the digital circuit may be automatically laid out. The physical standard-cell library may be linked to the layout tool **ICblock** so that the placement and routing of the cells used in the schematic can be done by **ICblock**. However, clock skew problems may be encountered if the clock signals are not distributed in balance.

After the layout design, the physical circuit should be simulated in a transistor-level simulator, like Mentor's **Lsim**. To facilitate the testing of the manufactured chips, a fault coverage analysis should be performed for the test vectors.

Bibliography

[1] B. P. Brandt and B. A. Wooley. “A 50-MHz multi-bit sigma-delta modulator for 12-b 2-MHz A/D conversion”. *IEEE J. Solid-State Circuits*, 26:1746–1756, Dec. 1991.

[2] H. Inose, Y. Yasuda, and J. Murakami. “A telemetering system code modulation—delta-sigmamodulation”. *IRE Trans. Space Elect. Telemetry*, SET-8:204–209, Sept. 1962.

[3] S. R. Norsworthy, R. Schreier, and G. C. Temes, editors. *“Delta-Sigma Data Converters: theory, design, and simulation”*. IEEE Press, New York, 1996.

[4] J. E. Franca and Y. Tsividis, editors. *“Design of Analog-Digital VLSI Circuits for Telecommunications and Signal Processing”*. Prentice Hall, Englewood Cliffs, New Jersey, 1994.

[5] D. A. Johns and K. Martin. *“Analog Integrated Circuit Design”*. Johns Wiley & Sons Inc., New York, 1996.

[6] K. M. Daugherty. *“Analog-to-Digital Conversion”*. McGraw-Hill Inc., New York, 1994.

[7] J. van Valburg and R. J. van de Plassche. “An 8-b 650-MHz folding ADC”. *IEEE J. Solid-State Circuits*, 27:1662–1666, Dec. 1992.

[8] K. Matsumoto, E. Ishii, K. Yoshitate, K. Amano, and R. W. Adams. “An 18-b oversampling A/D converter for digital audio”. *ISSCC Dig. Tech. Pap.*, pages 202–203, Feb. 1988.

[9] R. W. Adams. “Design and implementation of an audio 18-bit analog-to-digital converter using oversampling techniques”. *J. Audio Eng. Soc.*, 34:153–166, Mar. 1986.

[10] D. R. Welland, B. P. Del Signore, E. J. Swanson, T. Tanaka, K. Hamashita, S. Hara, and K. Takasuka. “A stereo 16-bit delta-sigma A/D converter for digital audio”. *J. Audio Eng. Soc.*, 37:476–486, June 1989.

[11] V. Friedman, D. M. Brinthaupt, D.-P. Chen, T. W. Deppa, J. P. Elward, E. M. Fields Jr., J. W. Scott, and T. R. Viswanathan. “A dual-channel voice-band PCM codec using $\sigma - \delta$ modulation technique”. *IEEE J. Solid-State Circuits*, SC-24:274–280, Apr. 1989.

[12] B. H. Leung, R. Neff, P. R. Gray, and R. W. Brodersen. “Area-efficent multichannel oversampled PCM voice-band coder. *IEEE J. Solid-State Circuits*, SC-23:1351–1357, Dec. 1988.

[13] L. Logo and M. Copeland. “A 13-bit ISDN-band oversampled ADC using two-stage third order noise shaping”. *IEEE Proc. Custom IC Conf.*, pages 21.2.1–21.2.4, Jan. 1988.

[14] R. Steele. *“Delta Modulation Systems”*. Johns Wiley & Sons Inc., New York, 1975.

[15] J. C. Candy and O. J. Benjamin. “The structure of quantization noise from sigma-delta modulation”. *IEEE Trans. Commun.*, COM-29:1316–1323, Sept. 1981.

[16] J. C. Candy. “A use of double integration in sigma-delta modulation”. *IEEE Trans. Commun.*, COM-33:249–258, Mar. 1985.

[17] R. W. Adams, P. F. Ferguson, 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.*, 39:515–528, July 1991.

[18] T. C. Leslie and B. Singh. "An improved sigma-delta modulator architecture". *IEEE Proc. ISCAS'90*, 1:372–375, May 1990.

[19] A. Hairapetian and G. C. Temes. "A dual-quantization multi-bit sigma-delta A/D converter". *IEEE Proc. ISCAS'94*, 5:437–440, May 1994.

[20] J. G. Proakis and D. G. Manolakis. *"Digital Signal Processing: principles, algorithms, and applications"*. Maxwell MacMillan Int., New York, 1992.

[21] A. V. Oppenheim. *"Discrete-time Signal Processing"*. Prentice Hall, Englewood Cliffs, N.J., 1989.

[22] R. E. Crochiere and L. R. Rabiner. "Interpolation and decimation of digital signals—A tutorial review". *Proc. IEEE*, pages 300–331, Mar. 1981.

[23] B. Brandt. *Oversampled Analog-to-Digital Conversion.* PhD thesis, Stanford University, 1991.

[24] L. A. Williams and B. A. Wooley. "Third-order cascaded sigma-delta modulators". *IEEE Tran. Circuits Syst.*, 38:489–498, May 1991.

[25] B. E. Boser and B. A. Wooley. "The design of sigma-delta modulation analog-to-digital converters". *IEEE J. Solid-State Circuits*, 23:1298–1308, Dec. 1988.

[26] K. Martin. "Improved circuits for the realization of switched-capacitor filters". *IEEE Trans. Circuits and Syst.*, CAS-27:237–244, Apr. 1980.

[27] Jr. P. Ferguson, A. Ganesan, and R. Adams. "An 18-b 20 khz dual ds a/d converter". *ISSCC Dig. Tech. Papers*, pages 68–69, Feb. 1991.

[28] T. Choi et al. “High-frequency cmos switched-capacitor filters for communications application”. *IEEE J. Solid-State Circuits*, pages 652–664, Dec. 1983.

[29] S. H. Lewis and P. R. Gray. “A pipelined 5-Msample/s 9-bit analog-to-digital converter”. *IEEE J. Solid-State Circuits*, pages 954–962, Dec. 1987.

[30] J. C. Candy. “Decimation for sigma delta modulation”. *IEEE Trans. Commun.*, pages 72–76, Jan. 1986.

[31] The MathWorks Inc. *“The Student Edition of MATLAB: Version 4 User’s Guide”*, 1995.

[32] A. Antoniou. *“Digital Filters : analysis, design, and applications”*. McGraw-Hill, New York, 1993.

[33] S. Chu and C. S. Burrus. “Multirate filter designs using comb filters”. *IEEE Trans. Circuits and Sys.*, pages 913–924, Nov. 1984.

[34] E. Dijkstra, O. Nys, C. Piguet, and M. Degrauwe. “On the use of modulo arithmetic comb filters in sigma delta modulators”. *IEEE Proc. ICASSP’88*, pages 2001–2004, Apr. 1988.

[35] J. Bhasher. *“A VHDL Primer: Revised Edition”*. Prentice Hall, Englewood Cliffs, NJ, 1994.

[36] Mentor Graphics Corp. *“The VHDL Style Guide For AutoLogic II”*, 1995.

[37] Mentor Graphics Corp. *“Synthesizing with AutoLogic II”*, 1996.

Appendix A

MOSIS SCMOS PROCESS

MOSIS is a public organization to supply ASIC fabrication services for world-wide IC designers. To apply MOSIS scaled-CMOS (SCMOS) technologies within Mentor Graphics tools, Mentor Design Kit (MDK) for MOSIS is needed for synthesis, layout, and simulation. The current version of MDK installed in the VLSI Lab is V1.7 which consists of SCMOS technology files, SCMOS standard cell libraries, and documentation. In this thesis work, I chose a $0.8-\mu m$ CMOS technology to implement the circuits.

A.1 SCN08HP Technology and Parameter File

Following is the technology and parameter file for $0.8 - \mu m$ SCMOS process available at Hewlett-Packard. This process is a CMOS N-well, 3-metal technology named *scn08hp* with lambda equal to $0.4\mu m$.

```
        // ICtrace Device Definitions
DEVICE C CAP POLY ELECTRODE (POS NEG) [0 0]
device r(dp) dpres psrcdrn psrcdrn [0.32]
device r(dn) dnres nsrcdrn nsrcdrn [0.368]


        // ICextract Rule Definitions
capacitance order POLY METAL1 METAL2 METAL3 direct
capacitance order NSRCDRN PSRCDRN POLY METAL1 METAL2 METAL3 mask
capacitance intrinsic METAL3 [0.0027 0]
capacitance intrinsic METAL2 [0.0035 0]
capacitance intrinsic METAL1 [0.0073 0]
capacitance intrinsic POLY [0.0137 0]
capacitance intrinsic PSRCDRN [0.0891 0.0257] mask
capacitance intrinsic NSRCDRN [0.048 0.0644] mask
capacitance crossover METAL3 METAL2 [0.0048 0 0]
capacitance crossover METAL3 METAL1 [0.0024 0 0]
capacitance crossover METAL3 POLY [0.00208 0 0]
capacitance crossover METAL2 METAL1 [0.00448 0 0]
capacitance crossover METAL2 POLY [0.00336 0 0]
capacitance crossover METAL1 POLY [0.0104 0 0]
resistance sheet METAL3 [0.008 0]
resistance sheet METAL2 [0.011 0]
resistance sheet METAL1 [0.011 0]
resistance sheet POLY [0.336 0]
resistance sheet PSRCDRN [0.32 0] mask
resistance sheet NSRCDRN [0.368 0] mask
resistance connection METAL1 POLY [0.48 0]
resistance connection METAL1 METAL2 [0.22 0]
resistance connection METAL2 METAL3 [0.16 0]
resistance connection METAL1 PSRCDRN [0.44 0] mask
resistance connection METAL1 NSRCDRN [0.52 0] mask
```

A.2 SPICE Model File

For SPICE simulation in Mentor's analog simulator AccuSim, a SPICE model file generated after a recent run was downloaded from MOSIS web site *http://www.isi.edu/mosis/vendors/hp-cmos26g.* The SPICE3 models for PMOS and NMOS are shown as follows:

```
    * FET models for MOSIS (HP 0.8 micron process on 3/24/97)
.MODEL n NMOS (PHI=0.700000 TOX=1.6600E-08 XJ=0.200000U TPG=1
+ VTO=0.7184 DELTA=6.3430E-01 LD=9.0910E-10 KP=1.2240E-04
+ UO=588.4 THETA=1.3630E-01 RSH=2.6960E+01 GAMMA=0.6249
+ NSUB=5.0910E+16 NFS=7.0710E+11 VMAX=1.9510E+05 ETA=3.5430E-02
+ KAPPA=1.2400E-01 CGDO=1.9000E-10 CGSO=1.9000E-10
+ CGBO=3.9199E-10 CJ=3.0900E-04 MJ=0.9500 CJSW=6.3000E-10
+ MJSW=0.33100 PB=0.700000)
.MODEL p PMOS (PHI=0.700000 TOX=1.6600E-08 XJ=0.200000U TPG=-1
+ VTO=-0.8434 DELTA=1.1570E+00 LD=9.4480E-10 KP=3.2971E-05
+ UO=158.5 THETA=1.2470E-01 RSH=5.0970E+01 GAMMA=0.4820
+ NSUB=3.0290E+16 NFS=5.9090E+11 VMAX=2.2220E+05 ETA=2.5980E-02
+ KAPPA=8.3010E+00 CGDO=1.9000E-10 CGSO=1.9000E-10
+ CGBO=4.0554E-10 CJ=5.5200E-04 MJ=0.4910 CJSW=1.83000E-10
+ MJSW=0.42500 PB=0.90000)
```

Appendix B

VHDL PROGRAMS

B.1 8-to-3 Encoder

```
-- Written by LL_to_VHDL at Thu Mar 20 21:42:03 1997
-- Parameterized Generator Specification to VHDL Code
--
-- LogicLib generator called: PRIORITY_ENCODER
-- Passed Parameters are:
--    tinst name = encoder
--    parameters are:
--       Priority = MSB
--       W = 8
--       WO = 3
--    program_values = (
--       0:000 1:001 2:010 3:011
--       4:100 5:101 6:110 7:111
--       d:000
--       )
--    Rows = 1
--
library IEEE, ARITHMETIC;
use IEEE.std_logic_1164.all;
use ARITHMETIC.std_logic_arith.all;

-- encoder Entity Description
```

```
entity encoder is
   port(
      DIN: in std_logic_vector(7 downto 0);
      DOUT: out std_logic_vector(2 downto 0)
   );
end encoder;

-- encoder Architecture Description
architecture rtl of encoder is
begin
   PRIORITY_ENCODER_Process: process(DIN)
      type enc_table is array (0 to 7) of std_logic_vector (2 downto 0);
      constant prio_table : enc_table :=
         ( "000", "001", "010", "011", "100", "101",
         "110", "111");
      variable enc_out : std_logic_vector(2 downto 0);
   begin
      enc_out := "000";
      chk_input: for x in DIN'reverse_range loop
         if DIN(x) = '1' then
           enc_out := prio_table(x);
         end if;
      end loop chk_input;
      -- Assign output
      DOUT <= enc_out;
   end process PRIORITY_ENCODER_Process;
end rtl;
```

B.2 Error Cancellation Logic

-- Error cancellation logic for 1-bit Y1 and 3-bit Y2

-- $Y(z) = (2z^{-2} - z^{-3})Y_1(z) - 4(1 - z^{-1})^2 Y_2(z)$

```
LIBRARY ieee, arithmetic;
use ieee.std_logic_1164.all;
use arithmetic.std_logic_arith.all;

entity e_cancel is
   generic(multi_bit: integer :=4);
   port( y1:in std_logic:='0'; clk: in bit := '0';
      y2: in signed (multi_bit - 2 downto 0):=(others=>'0');
      y: out signed (multi_bit downto 0));
end e_cancel;

architecture behav of e_cancel is

   signal y1_d1, y1_d0: std_logic:='0';
   signal y2_d1, y2_d0, sum1, sum1_d: signed (multi_bit downto 0):= (others=>'0');
   signal sum2, y1_d2, y1_d3, temp: signed (multi_bit downto 0):= (others => '0');

begin

   sum1<=y2_d0 - y2_d1;
   temp<=(sum1 sla 1);
   sum2<=((y1_d2 sla 1) - y1_d3) - (temp - sum1_d);
   process(clk)
   begin
      if (clk='1' and clk'event) then
        y1_d0<=y1;
        y2_d0<=y2(y2'left) & y2 & '0';
        y1_d1<=y1_d0;
        y2_d1<=y2_d0;
        for i in (y1_d2'left - 1) downto 0 loop
           y1_d2(i)<=y1_d1;
        end loop;
        y1_d2(y1_d2'left)<=not y1_d1;
        y1_d3<=y1_d2;
        sum1_d<=temp;
        y<=sum2;
      end if;
   end process;

end architecture behav;
```

B.3 Comb Filter

-- Decimator using $sinc^4$ filter

-- internal word length $l \geq \lg(d * (n^k + 1))$

-- d=input word length, n=decimation ratio, k=order of sinc

```
LIBRARY ieee, arithmetic;
use ieee.std_logic_1164.all;
use arithmetic.std_logic_arith.all;

entity decimator is
   generic(d:integer:=5; k: integer:=4; l:integer:=13);
   port( clk1, clk2, rst: in bit := '0';
      x: in signed (d - 1 downto 0):=(others=>'0');
      y: out signed (l -1 downto 0)); end decimator;

architecture opt of decimator is

   type sinc_adder is array (integer range <>) of signed (l -1 downto 0);
   signal sum_d, diff: sinc_adder(0 to k);
   signal sum, diff_d: sinc_adder(0 to k -1);

begin

   sum_d(0)(x'left downto 0)<=x;
   sum_d(0)(sum_d(0)'left downto x'left +1)<=(others=>x(x'left));
   gk: for i in 0 to (k -1) generate
      sum(i)<=sum_d(i) + sum_d(i +1);
      diff(i +1)<=diff(i) -diff_d(i);
   end generate gk;

   process(clk1, clk2)
   begin
      if (clk1='1' and clk1'event) then
        if (rst='1') then
```

```
            fori in 0 to k loop
               sum_d(i)<=(others=>'0');
            end loop;
          else
            for i in 1 to k loop
               sum_d(i)<=sum(i -1);
            end loop;
          end if;
        end if;
        if (clk2='1' and clk2'event) then
          diff(0)<=sum_d(k);
          for i in 0 to (k - 1) loop
            diff_d(i)<=diff(i);
          end loop;
          y<=diff(k);
        end if;
    end process;

end architecture opt;
```

B.4 Half-Band FIR Filters

```
-- 11 taps and 165 taps half band FIR filters


library ieee, arithmetic;
use ieee.std_logic_1164.all;
use arithmetic.std_logic_arith.all;
```

```
entity fir is
   generic(l:integer:=13);
   port(clk1, clk2:in bit:='0'; x: in signed(l -1 downto 0):=(others=>'0');
      y:out signed(l -1 downto 0));
end fir;

architecture hbf11 of fir is

   type data_bus is array (integer range <>) of signed (l -1 downto 0);
   signal x_d:data_bus(0 to 10);
   signal sum: data_bus(0 to 2);
   signal sum_d: data_bus(0 to 3);

begin
   shift: process (clk1, clk2)
   begin
      if (clk1='1' and clk1'event) then
        for i in 1 to 10 loop x_d(i)<=x_d(i -1);
        end loop;
      end if;
      if (clk2='1' and clk2'event) then
        sum_d(0)<=(sum(0) sra 7) + (sum(0) sra 8) + (sum(0) sra 11);
        sum_d(1)<=(sum(1) sra 4);
        sum_d(2)<=((sum(2) sra 2) + (sum(2) sra 5)) + ((sum(2) sra 6) + (sum(2) sra 8));
        sum_d(3)<=(x_d(5) sra 1);
        y<=(sum_d(0) - sum_d(1)) + (sum_d(2) + sum_d(3));
      end if;
   end process shift;

   x_d(0)<=x;
   add: for i in 0 to 2 generate
      sum(i)<=x_d(2 * i) + x_d(10 - 2 * i);
   end generate add;

end architecture hbf11;


architecture hbf165 of fir is

   type data_bus is array (integer range ¡¿) of signed (l -1 downto 0);
   signal x_d:data_bus(0 to 163);
   signal sum: data_bus(0 to 40);
   signal sum_d: data_bus(0 to 20);

begin
   shift: process (clk1, clk2)
   begin
      if (clk1='1' and clk1'event) then
        for i in 1 to 163 loop x_d(i)<=x_d(i -1);
        end loop;
      end if;
      if (clk2='1' and clk2'event) then
        sum_d(0)<=(sum(0) sra 11) - (sum(1) sra 12);
sum_d(1)<=(sum(2) sra 12) - (sum(3) sra 11);
sum_d(2)<=(sum(4) sra 11) - (sum(5) sra 11);
sum_d(3)<=((sum(6) sra 11) + (sum(6) sra 12)) - (sum(7) sra 10);
sum_d(4)<=(sum(8) sra 10) - (sum(9) sra 10);
sum_d(5)<=(sum(10) sra 9) - (sum(11) sra 9);
sum_d(6)<=(sum(12) sra 9) - (sum(13) sra 9);
sum_d(7)<=((sum(14) sra 9) + (sum(14) sra 12)) -
          ((sum(15) sra 9) + (sum(15) sra 11));
sum_d(8)<=((sum(16) sra 9) + (sum(16) sra 10)) -
          ((sum(17) sra 9) + (sum(17) sra 10));
sum_d(9)<=(sum(18) sra 8) - (sum(19) sra 8);
sum_d(10)<=((sum(20) sra 8) + (sum(20) sra 11)) -
          ((sum(21) sra 8) + (sum(21) sra 10));
sum_d(11)<=((sum(22) sra 8) + (sum(22) sra 9)) -
          ((sum(23) sra 8) + (sum(23) sra 9));
sum_d(12)<=(sum(24) sra 7) - (sum(25) sra 7);
sum_d(13)<=((sum(26) sra 7) + (sum(26) sra 11)) -
          ((sum(27) sra 7) + (sum(27) sra 9));
sum_d(14)<=(((sum(28) sra 7) + (sum(28) sra 9)) + (sum(28) sra 10)) -
          ((sum(29) sra 7) + (sum(29) sra 8));
sum_d(15)<=((sum(30) sra 7) + (sum(30) sra 8)) + ((sum(30) sra 9) -
          (sum(31) sra 6));
sum_d(16)<=((sum(32) sra 6) + (sum(32) sra 9)) -
          ((sum(33) sra 6) + (sum(33) sra 8) + (sum(33) sra 12));
sum_d(17)<=((sum(34) sra 6) + (sum(34) sra 7)) - (((sum(35) sra 6) +
          (sum(35) sra 7)) + ((sum(35) sra 8)+ (sum(35) sra 11)));
sum_d(18)<=((sum(36) sra 5) + (sum(36) sra 8)) -
```

```
                    ((sum(37) sra 5) + (sum(37) sra 6));
        sum_d(19)<=(sum(38) sra 4) - (((sum(39) sra 4) + (sum(39) sra 5)) +
                    ((sum(39) sra 7) + (sum(39) sra 8)));
        sum_d(20)<=(x_d(5) sra 1) + (((sum(40) sra 2)+(sum(40) sra 4)) +
                    ((sum(40) sra 8) + (sum(40) sra 9)));
        y<=(((sum_d(0) + sum_d(1)) + (sum_d(2) + sum_d(3))) +
                    ((sum_d(4) + sum_d(5)) + (sum_d(6) + sum_d(7)))) +
                    (((sum_d(8) + sum_d(9)) + (sum_d(10) + sum_d(11))) +
                    ((sum_d(12) + sum_d(13)) + (sum_d(14) + sum_d(15)))) +
                    (((sum_d(16) + sum_d(17)) + (sum_d(18) + sum_d(19))) +
                    sum_d(20));
      end if;
   end process shift;

   x_d(0)<=x;
   add: for i in 0 to 40 generate
      sum(i)<=x_d(2 * i +1) + x_d(163 - 2 * i);
   end generate add;

end architecture hbf165;
```

B.5 Structure Modeling for Digital Circuit

```
-- digital filter for delta-sigma converter

LIBRARY ieee, arithmetic;
use ieee.std_logic_1164.all;
use arithmetic.std_logic_arith.all;

entity filter is
   generic(multi_bit: integer :=4; l: integer:=12; n: integer:=24);
   port( x1: in std_logic:='0'; clk, rst: in bit := '0';
      x2: in signed (multi_bit - 2 downto 0):=(others=>'0');
      y: out signed (l -1 downto 0));
end filter;

architecture struc of filter is

   signal y1, y2, y3: signed (l downto 0):=(others=>'0');
   signal x: signed (multi_bit downto 0):= (others => '0');
   signal clk1, clk2, clk3: bit:='0';
   signal count: integer range 0 to (n -1):=0;

begin

   counter: process(clk)
   begin
      if (clk='1' and clk'event) then
        if (rst='1') then
          count<=0;
        else
          if (count =n -1) then count<=0;
          else count<=count +1;
          end if;
```

```
            end if;
        end if;
    end process counter;

    clk_gen: block
    begin
        clk1<='1' when(count=0 or count=(n /4) or count=(n /2) or
             count=(3 *n /4)) else '0';
        clk2<='1' when(count=0 or count=(n /2)) else '0';
        clk3<='1' when(count=0) else '0';
    end block clk_gen;

    i1: entity work.e_cancel(behav) port map(x1, clk, x2, x);
    i2: entity work.decimator(opt) port map(clk, clk1, rst, x, y1);
    i3: entity work.fir(hbf11) port map(clk1, clk2, y1, y2);
    i4: entity work.fir(hbf165) port map(clk2, clk3, y2, y3);
    y<=y3(l downto 1);

end architecture struc;
```

Appendix C

MATLAB PROGRAMS

C.1 Herrmann Estimation of FIR Filter Order

```
% Formula proposed by Herrmann for approximating the length
% of the linear phase FIR filter.

function m=herr(sf, pc, sc, pr, sr)

% sf: sampling frenquency
% pc: passband cutoff frequency
% sc: stopband cutoff frequency
% pr: passband ripple
% sr: stopband ripple

x=(sc-pc)/sf;
d=(0.005309*(log10(pr))^2 + 0.07114*log10(pr) - 0.4761)*log10(sr) ...
  - (0.00266*(log10(pr))^2 + 0.5941*log10(pr) + 0.4278);
f=11.012 + 0.51244*(log10(pr) - log10(sr));
m=(d - f*x^2)/x + 1;
```

C.2 Frequency Response of $sinc^k$ Comb Filters

```
% frequency response for a sinc^k comb filter

function [yy, h, ny, pd]=comb(m, k, f, r)
```

```
% m=decimation ratio, k=order of sinc, f=frequency in Hz
% r=oversampling ratio

% construct the coefficient vector for $-z^{-m}+1$
x=1:(m+1);
x(1)=-1;
x(m+1)=1;
fori=2:m
   x(i)=0;
end
% the coefficient vector for $-z^{-1}+1$
y=-1:2:1;
% the coefficient vectors for $(-z^{-m}+1)^k$ and $[m(-z^{-1}+1)]^k$
nd=conv(x, x); % function conv multiplies two polynomials.
dd=conv(y, y);
for i=1:(k-2)
   nd=conv(nd, x);
   dd=conv(dd, y);
end
nd=fliplr(nd); % flip vectors left-right so that theses vectors may be
dd=fliplr(dd); % used in function freqz.
dd=dd*(m^k);
% frequency response of a digital filter given the numerator
% and denominator coefficients in vectors nd and dd.
ny=0:(f/(512*r)):(f/r); % 512-point frequencies in baseband of Nyquist rate.
[h, p]=freqz(nd, dd, 512, f); % frequency response between 0 and f/2.
```

```
pd=freqz(nd, dd, ny, f); % frequency response in range of 0–f/r.
h=20*log10(abs(h)); % magnitude in dB.
pd=20*log10(abs(pd));
yy=1:512;
for i=1:512
   yy(i)=(i-1)*f/1024; % y-axis vector of frequency.
end

% plot the bode diagrams.
subplot(2,1,1);
plot(yy,h);
title('Comb Filter Frequency Response');
grid on;
ylabel('Magnitude (dB)');
xlabel('Frequency (Hertz)');
subplot(2,1,2);
plot(nyquist, fr);
%title('Magnitude Decrease in Passband');
grid on;
ylabel('Magnitude (dB)');
xlabel('Frequency (Hertz)');
```

C.3 Two-Stage Half-Band Filter Design

Figure 4.3 is generated by calling the following function *half_band* in Matlab as `half_band(1, 50/6, 10, 164)`, where 1 corresponds to 1 MHz passband cutoff frequency, 50/6 indicates the sampling rate of 50/6 = 8.33 MHz for the input signal of the half-band filters, and 10 and 164 correspond to orders of the 11-tap and 165-tap filters, respectively.

```
% Calculation of coefficients for two cascaded half band filters
% fp=pass band in MHz, fs=initial sampling frequency in MHz
% n1=order of the first filter (n-tap filter is (n-1)th-order)
% n2=order of the second filter
% n1 and n2 should be even numbers for odd number of taps (coefficients)

function [b1, b2]=half_band(fp, fs, n1, n2)

f1=[0 2*fp/fs 1-2*fp/fs 1]; % pairs of frequency points between 0 and 1
m=[1 1 0 0]; % the desired magnitude response at the points specified in f
f2=[0 4*fp/fs 1-4*fp/fs 1];
b1=remez(n1, f1, m); % remez generates equiripple FIR filter
b2=remez(n2, f2, m);

[h1, w1]=freqz(b1, 1, 512, fs*10^6); % frequency response
[h2, w2]=freqz(b2, 1, 512, fs*10^6/2);
h1=20*log10(abs(h1)); % calculating in dB
h2=20*log10(abs(h2));

% plot the Bode diagrams
subplot(2,2,1);
y=1:512;
fori=1:512
  y(i)=(i-1)*fs*10^6/1024;
end
```

```
plot(y, h1);
title('The First Half Band Filter');
grid on;
axis([0 fs/2*10^6 -110 10]);
ylabel('Magnitude (dB)');
xlabel('Frequency (Hertz)');
subplot(2,2,2);
for i=1:512
  y(i)=(i-1)*fs*10^6/2048;
end
plot(y, h2);
title('The Second Half Band Filter');
grid on;
axis([0 fs/4*10^6 -110 10]);
ylabel('Magnitude (dB)');
xlabel('Frequency (Hertz)');

[h1, w1]=freqz(b1, 1, 512);
[h2, w2]=freqz(b2, 1, 512);

% plot the nomalized diagrams
subplot(2,2,3);
plot(f1, m, '-', w1/pi, abs(h1), '.'); % dashed curve is the ideal response.
grid on;
axis([0 1 -0.1 1.1]);
ylabel('Magnitude');
xlabel('Frequency');
```

```
subplot(2,2,4);
plot(f2, m, '-', w2/pi, abs(h2), '.'); % dotted curve is the designed response.
grid on;
axis([0 1 -0.1 1.1]);
ylabel('Magnitude');
xlabel('Frequency');
```

C.4 Half-Band Filters with Rounded Coefficients

Figure 4.4 is generated by calling `hbf1(1, 50/6)`, while Figure 4.5 is generated by calling `hbf2(1, 50/12)`. The rounded coefficients are listed in Table 4.2.

```
% Frequency response of 11-tap half band filter with rounded coefficients
% fp=pass band in MHz, fs=sampling frequency in MHz

function b1=hbf1(fp, fs)

f1=[0 2*fp/fs 1-2*fp/fs 1];
m=[1 1 0 0];
% construct vector for rounded coefficients
b1= [2^(-7)+2^(-8)+2^(-11) 0 -2^(-4) 0 2^(-2)+2^(-5)+2^(-6)+2^(-8) 0.5 ...
    2^(-2)+2^(-5)+2^(-6)+2^(-8) 0 -2^(-4) 0 2^(-7)+2^(-8)+2^(-11)];
[h1, w1]=freqz(b1, 1, 512, fs*10^6);
h1=20*log10(abs(h1)); % take logarithm of the magnitude

subplot(2,2,1);
y=1:512;
for i=1:512
```

```
    y(i)=(i-1)*fs*10^6/1024;
end
plot(y, h1);
title('Log Magnitude');
grid on;
axis([0 fs/2*10^6 -100 10]);
ylabel('Magnitude (dB)');
xlabel('Frequency (Hertz)');

subplot(2,2,3);
plot(y, h1);
title('Passband Ripples');
grid on;
axis([0 fp*10^6 -0.015 0.015]); % passband 0-fp MHz
ylabel('Magnitude (dB)');
xlabel('Frequency (Hertz)');

[h1, w1]=freqz(b1, 1, 512);
subplot(2,2,2);
plot(f1, m, '-', (w1/pi), abs(h1), '.');
title('Nomalized Magnitude');
grid on;
axis([0 1 -0.1 1.1]);
ylabel('Magnitude');
xlabel('Frequency');

subplot(2,2,4);
```

```
plot(f1, m, '.', (w1/pi), abs(h1), '-');
title('Stopband Ripples');
grid on;
axis([1-2*fp/fs 1 -0.0005 0.0015]); % stopband from 1-fp/(fs/2) to 1
ylabel('Magnitude');
xlabel('Frequency');


% frequency responses of the 165-tap half-band filter with rounded coefficients
% fp=pass band in MHz, fs=sampling frequency in MHz

function b=hbf2(fp, fs)

f1=[0 2*fp/fs 1-2*fp/fs 1];
m=[1 1 0 0];
b=remez(164, f1, m);
fori=1:2:165
  b(i)=0;
end
% rounded coefficients
b(2)=2^(-11);
b(4)=-2^(-12);
b(6)=2^(-12);
b(8)=-2^(-11);
b(10)=2^(-11);
b(12)=-2^(-11);
b(14)=2^(-11)+2^(-12);
```

```
b(16)=-2^(-10);

b(18)=2^(-10);

b(20)=-2^(-10);

b(22)=2^(-9);

b(24)=-2^(-9);

b(26)=2^(-9);

b(28)=-2^(-9);

b(30)=2^(-9)+2^(-12);

b(32)=-(2^(-9)+2^(-11));

b(34)=2^(-9)+2^(-10);

b(36)=-(2^(-9)+2^(-10));

b(38)=2^(-8);

b(40)=-2^(-8);

b(42)=2^(-8)+2^(-11);

b(44)=-(2^(-8)+2^(-10));

b(46)=2^(-8)+2^(-9);

b(48)=-(2^(-8)+2^(-9));

b(50)=2^(-7);

b(52)=-2^(-7);

b(54)=2^(-7)+2^(-11);

b(56)=-(2^(-7)+2^(-9));

b(58)=2^(-7)+2^(-9)+2^(-10);

b(60)=-(2^(-7)+2^(-8));

b(62)=2^(-7)+2^(-8)+2^(-9);

b(64)=-2^(-6);

b(66)=2^(-6)+2^(-9);

b(68)=-(2^(-6)+2^(-8)+2^(-12));
```

```
b(70)=2^(-6)+2^(-7);
b(72)=-(2^(-6)+2^(-7)+2^(-8)+2^(-11));
b(74)=2^(-5)+2^(-8);
b(76)=-(2^(-5)+2^(-6));
b(78)=2^(-4);
b(80)=-(2^(-4)+2^(-5)+2^(-7)+2^(-8));
b(82)=2^(-2)+2^(-4)+2^(-8)+2^(-9);
b(83)=2^(-1);
for i=2:2:82
  b(166-i)=b(i);
end
[h1, w1]=freqz(b, 1, 512, fs*10^6);
h1=20*log10(abs(h1));

subplot(2,2,1);
y=1:512;
for i=1:512
  y(i)=(i-1)*fs*10^6/1024;
end
plot(y, h1);
title('Log Magnitude');
grid on;
axis([0 fs/2*10^6 -100 10]);
ylabel('Magnitude (dB)');
xlabel('Frequency (Hertz)');

subplot(2,2,3);
```

```
plot(y, h1);
title('Passband Ripples');
grid on;
axis([0 fp*10^6 -0.1 0.1]);
ylabel('Magnitude (dB)');
xlabel('Frequency (Hertz)');

[h1, w1]=freqz(b, 1, 512);
subplot(2,2,2);
plot(f1, m, '-', w1/pi, abs(h1), '.');
title('Nomalized Magnitude');
grid on;
axis([0 1 -0.1 1.1]);
ylabel('Magnitude');
xlabel('Frequency');

subplot(2,2,4);
plot(f1, m, '.', w1/pi, abs(h1), '-');
title('Stopband Ripples');
grid on;
axis([1-2*fp/fs 1 -0.005 0.015]);
ylabel('Magnitude');
xlabel('Frequency');
```