A BIOLOGICALLY INSPIRED FRONT END FOR AUDIO SIGNAL PROCESSING USING PROGRAMMABLE ANALOG CIRCUITRY

A Dissertation
Presented to
The Academic Faculty

By

David W. Graham

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in Electrical Engineering

School of Electrical and Computer Engineering
Georgia Institute of Technology
August 2006

A BIOLOGICALLY INSPIRED FRONT END FOR AUDIO SIGNAL PROCESSING USING PROGRAMMABLE ANALOG CIRCUITRY

Approved by:

Dr. Paul Hasler, Advisor
Associate Professor, School of ECE
Georgia Institute of Technology

Dr. Aaron Lanterman
Assistant Professor, School of ECE
Georgia Institute of Technology

Dr. David V. Anderson
Associate Professor, School of ECE
Georgia Institute of Technology

Dr. W. Marshall Leach
Professor, School of ECE
Georgia Institute of Technology

Dr. F. Levent Degertekin
Associate Professor, School of ME
Georgia Institute of Technology

Date Approved: June 13, 2006

To my wife Meadow.

Thank you for giving me up to long hours and late nights in lab.

ACKNOWLEDGMENTS

I would like to thank my parents for their continued love and support throughout all my life and all my academic endeavors. Without them, I would not have made it this far.

I would especially like to thank Dr. Paul Hasler, my advisor. He has been a constant source of help in my academic endeavors and has prepared me well for my future career.

I would also like to thank Dr. David Anderson, Dr. F. Levent Degertekin, Dr. Aaron Lanterman, and Dr. W. Marshall Leach who are the other members of my thesis committee. They have helped to give me great ideas and new ways to approach my research.

Everyone in the Cooperative Analog-Digital Signal Processing (CADSP) group has been wonderful. Thank you for all the help in understanding difficult concepts, help in debugging circuits, and the friendships we have formed.

I would like to extend a special thanks to Analog Devices, with which a valuable internship was available through the Georgia Tech Analog Consortium. This work was partially funded by generous funding from the National Science Foundation and the Hewlett-Packard Company.

Without my wife, Meadow, I certainly would have had a much harder time making it through this academic journey. She has been the best friend anyone could ever imagine and a constant source of strength.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

This research focuses on biologically inspired audio signal processing using programmable analog circuitry. This research is inspired by the biology of the human cochlea since biology far outperforms any engineered system in converting audio signals into meaningful electrical signals. The human cochlea efficiently decomposes any sound into the respective frequency components by harnessing the resonance nature of the basilar membrane, essentially forming a bank of bandpass filters. In a similar fashion, this work revolves around developing a filter bank composed of continuous-time, low-power, analog bandpass filters that serve as the core front end to this silicon audio-processing system. Like biology, the individual bandpass filters are tuned to have narrow bandwidths, moderate amounts of resonance, and exponentially spaced center frequencies. To overcome mismatch and offsets inherent in CMOS processes, floating-gate transistors are used to precisely tune the time constants in the filters and to allow programmability of analog components.

Like the biological cochlea, this audio front end serves to efficiently convert incoming sounds into information useful to the subsequent signal-processing elements, and it does so by performing a frequency decomposition of the waveform with extremely low-power consumption and real-time operation. This frequency decomposition can be used to replace discrete Fourier transforms which are expensive computationally and consume large amounts of power. As portable electronics progressively pervade everyday life, power constraints become increasingly important; the power savings of this analog frequency-decomposition block will be able to greatly extend battery life in consumer electronics and embedded sensors, such as hearing aids and cohclear implants. Additionally, the floating-gate programmability of this filter bank, especially when incorportated in a reconfigurable architecture, will allow versatility so that a variety of algorithms can be produced by the same integrated circuit.

CHAPTER 1

INTRODUCTION

Biological systems far outperform any engineered system at perceiving the outside world and making useful decisions based upon those perceptions. Biology is able to perform these immensely complex perception and classification tasks at real-time speeds yet only consume the little power that the body can provide. I believe that by looking to biology for inspiration and by developing solutions that mimic biological structures, engineers will be able to develop improved solutions to a variety of problems that will reduce power consumption, operate at real-time speeds, and perform better than present techniques. Thus, in my research, I am focusing on developing bio-inspired analog circuits and systems for signal-processing applications.

The human auditory system is capable of astonishing feats of recognizing words, adapting to different sound levels, and localizing sounds, all of which are difficult tasks to engineer well. However, the ability to incorporate these tasks into portable electronics would revolutionize the marketplace. The market for portable audio devices (e.g. music players and cell phones) is steadily increasing, but standard methods for performing the complex auditory algorithms consume far too much power and need too much overhead for inexpensive portable applications. If these tasks could be performed in real time and at low power, like the human ear, then one could capitalize on the growing market for portable audio devices. Additionally, by looking to biology for inspiration and developing solutions that mirror biological structures, it will be feasible to develop portable audio devices that perform better than current systems while working at a fraction of the presently required power.

When developing biologically inspired hardware, performing at least a portion of the computation in the analog domain is advantageous because analog circuitry readily performs biology-like computations, whereas these same computations can

be burdensome with digital circuitry [1]. Additionally, analog circuitry typically consumes far less power than digitaly circuitry and will, thus, allow more signal processing to be performed for a given power budget.

1.1 Analog Signal Processing Advantages

The tendency in the signal-processing realm for dealing with signals coming directly from an audio sensor is to immediately pass the inherently analog signal to an analog-to-digital converter (ADC) so that the signal can be manipulated digitally. Typically, a Fourier transform, such as a discrete- or fast-Fourier transform (DFT or FFT), of the signal is performed digitally so that the individual subbands can be manipulated. Digital signal processing (DSP) is invoked as early as possible since it has many advantages, and the greatest is the ease of programming a digital system to meet the given requirements.

However, another option is to introduce an analog system that does more than simply convert an audio signal into a digital version as soon as possible. By placing an analog signal-processing (ASP) block immediately after the audio sensor and immediately before an ADC, as is shown in Figure 1a, much of the processing can be done with the low-power and real-time computation of analog circuitry. This ASP block, therefore, alleviates a large portion of the digital circuitry's burden. The overall system can either have a smaller digital processing block than was previously required, or it can have the same size digital block, thus allowing for more functionality since the basic processing has already been conducted in analog.

Many options exist for an analog signal-processing block. However, if frequency decomposition is possible with analog circuitry, then this is a clear choice for the front-end analog block. In addition, more signal processing could be performed with analog circuits on the subband signals before they are recombined or sent through individual, smaller ADCs, as is illustrated in Figure 1b.

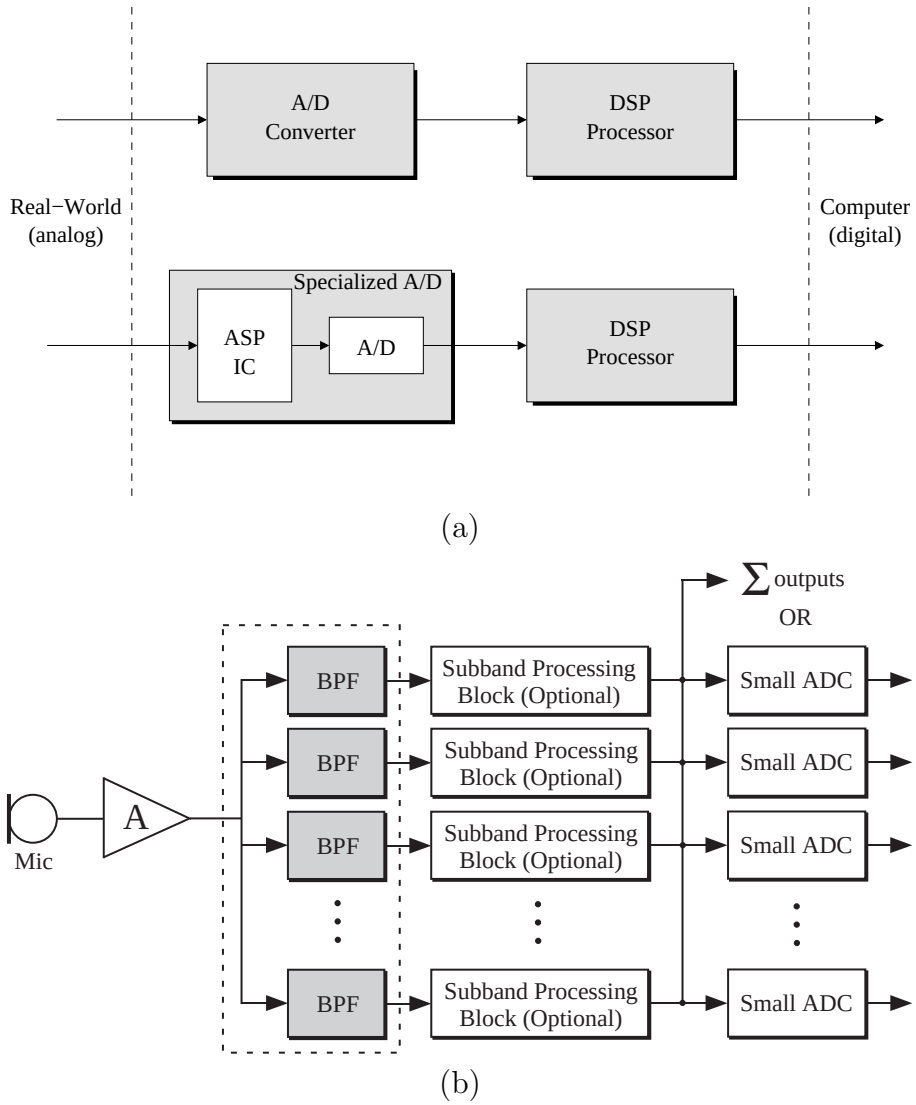


Figure 1. (a) Collaborative analog and digital signal processing blurs the boundary of where to place the conversion from analog to digital. Performing some of the signal processing with low-power, real-time analog circuitry alleviates some of the burden of the DSP allowing the DSP to perform more complex computations or a smaller DSP to be used. (b) The analog signal-processing block presented in this document, which can be used as a smart sensor interface, consists of an array of programmable bandpass filters (shown in the dashed box). Further signal processing can be performed on each subband signal before either recombining them or sending them through small ADCs and then on to the DSP.

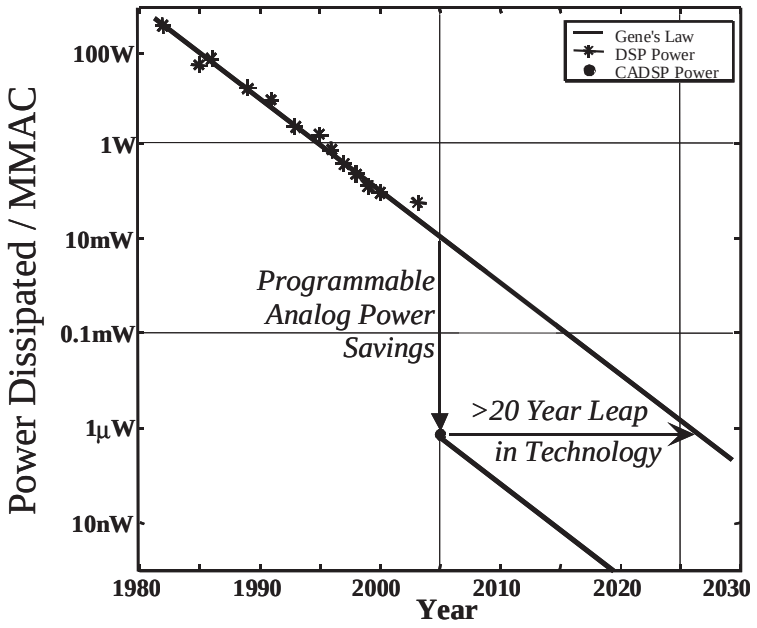


Figure 2. Power savings of analog over digital. DSP hardware power consumption follows the trend given by Gene's Law [2]. Analog computational blocks provide incredible power savings compared to digital counterparts, which is equivalent to a 20-year leap in technology.

1.2 Analog Frequency Decomposition Power Savings

One of the primary reasons for using an analog block to perform a frequency decomposition is the reduced power consumption as compared to a digital system. In fact, analog circuitry biased in the subthreshold regime leads to greatly reduced power consumption over digital processing for a wide variety of tasks. Comparing the trends in DSP hardware power consumption to analog circuitry's power consumption shows that a significant power savings exists by using analog circuitry, as is shown in Figure 2. Extrapolating out the DSP power consumption trend by following Gene's Law [2], we see that the power savings to perform an analog equivalent of a multiply-accumulate (MAC) is equivalent to a 20-year leap in technology [3].

Computing an estimate of the power savings associated with performing a frequency decomposition in analog can be computed as follows. Using the analytic expression for the power consumed by each individual bandpass filter as given by (31),

which is presented in Chapter 3, an entire filter bank with 32 subbands consumes $\leq 5\mu$W.

To achieve a similar level of performance digitally using an FFT with 32 subbands and operating at 44.1kHz requires approximately 50MMACS. Using some of the most power efficient DSPs on the market that operate at $4 - 10$MMACS/mW [4], similar computations require approximately 5mW. The analog block, therefore, yields a power savings on the order of a factor of 1000.

Because of the very low-power nature of this analog bandpass array, a spectrum of opportunies are available, including opportunities in high-quality hearing aids which aim to operate with no more than 1mW of power. Using this analog filter bank allows tremendously more signal processing to be conducted, even in the digital domain, so that complex algorithms may be implemented while still meeting the power budget of a hearing aid.

1.3 Signal-to-Noise Ratio Versus Cost

Even if analog circuitry is able to provide the same functionality as digital circuitry, an important question is how well this functionality can be achieved in the analog domain. One way to determine this answer is to find the effective resolution of comparable analog and digital systems, or, similarly, identify the cost of computation for a particular resolution.

Figure 3a shows a typical plot of signal-to-noise ratio (SNR) as bits of resolution versus the net cost, where the cost encompasses a wide range of metrics including area, power dissipation, computational delay, required tools, expenses associated with the design and manufacture, and design time [5]. The cost of digital computation varies linearly with the number of bits of resolution, while the cost of analog computation, which uses only a single pathway as opposed to a bus as required in digital systems, varies exponentially with the number of bits of resolution. As a result, computation

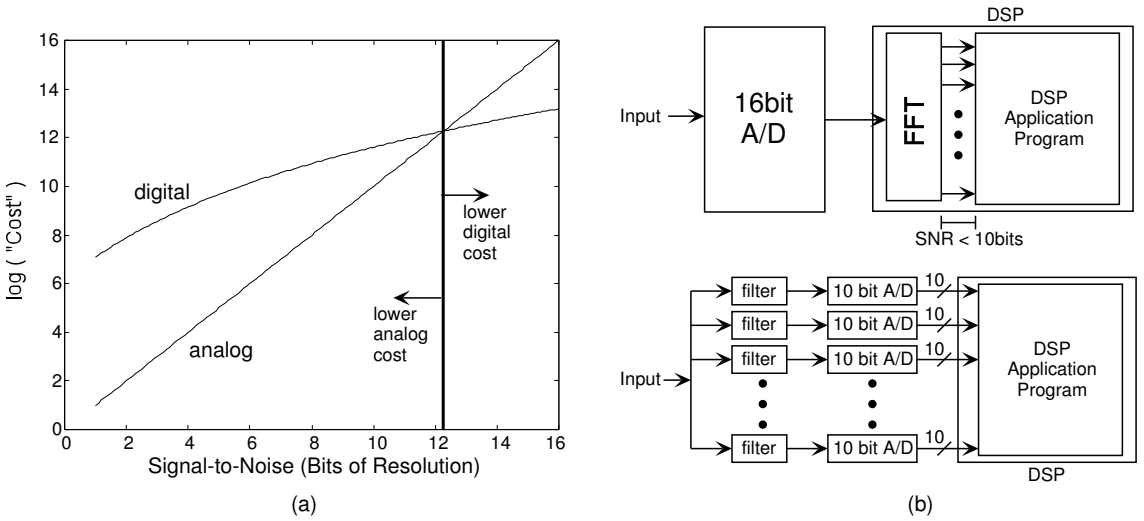


Figure 3. Analog and digital design cost with respect to performance. (a) The computational cost associated with the design of analog circuitry yields better performance in terms of signal-to-noise ratio than does digital design below a given threshold. (b) An example system for audio applications showing the purely digital solution and also the combined analog and digital solution.

requiring less resolution than a specific threshold is less expensive for analog computation, but above this same threshold, computation is less expensive in the digital domain. This threshold is typically between 8-14 bits [6].

The key in determining the necessary resolution for either the analog or digital parts depends heavily on the amount of the incoming information and the resolution needed to represent it, and this concept is illustrated in Figure 3b. As was previously stated, for audio systems, the analog waveform coming directly from the microphone is typically converted into a digital signal immediately. This raw digital waveform is then broken into frequency subbands via a discrete-Fourier transform, and then further subband processing can be performed digitally. Alternatively, this frequency decomposition can be done in the analog domain and then converted to the digital domain for further subband processing via multiple ADCs or a single multiplexed ADC, as is shown in Figure 3b.

Both analog portions have similar design complexity; the design complexity of 16-bit ADCs is exponentially more difficult than the design complexity of 10-bit ADCs.

When determining ASP resolution, typically measured in SNR, the particular circuit effects and continuous-time signal processing must be accounted for in order to attain an accurate estimate. Simply treating analog components as fixed-point arithmetic with finite-register effects will always underestimate the SNR of actual computation.

In general, performing more tasks in digital hardware generally increases flexibility and increases power consumption and, beyond a certain point, can yield increased accuracy. However, analog implementations of parts of a system normally result in significant power and space savings at the expense of flexibility. Flexibility wihtin analog systems, however, will be shown to be achievable in Chapter 7 with the use of floating-gate transistors.

1.4 An Example System and Related Applications

An example composite system illustrating the use of an analog block performing frequency decomposition is shown in Figure 4. In this system, the frequency decomposition interfaces directly with the audio sensor and allows for either analog or digital processing to be performed on each frequency subband. Such a system could be envisioned to be very useful in an embedded sensor in which a high degree of performance is required at very low levels of power consumption.

This system consists of a microphone, a low-noise amplifer (LNA), and an array of bandpass filters. The microphone and LNA are combined into a single capacitive sensor, as illstrated in Figure 4 and described in detail in [7]. A MEMS microphone, serving as the variable capacitor, is connected to a floating-gate amplifier in a charge-amplifier configuration, which serves as the LNA. Therefore, the amplifier senses the change in capacitance that is caused by acoustical vibrations, then transduces those fluctuations into an electrical signal, and finally amplifies the resulting electrical signal so that the filter bank can operate on it. Early measurements [7] have shown that this capacitive sensor operates very cleanly (output-referred noise $\leq 1\mu\text{V}/\sqrt{\text{Hz}}$)

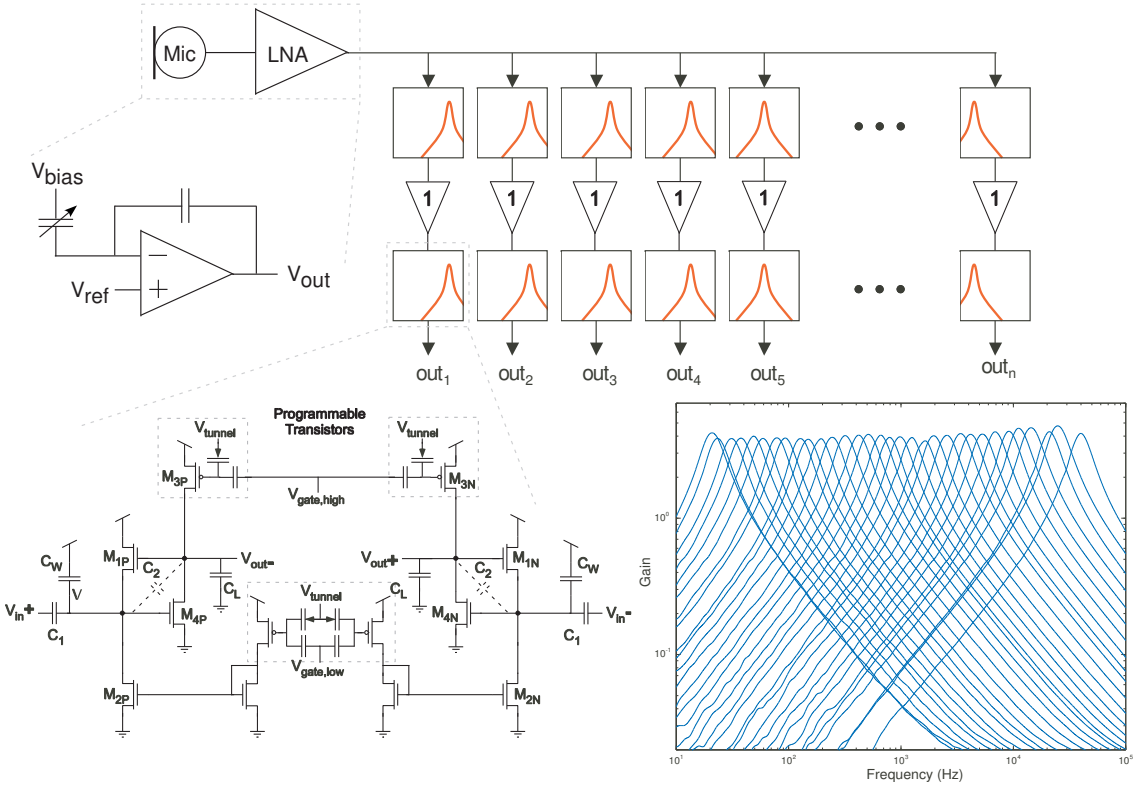


Figure 4. An entire system for an embedded smart audio sensor. The microphone and low-noise amplifier (LNA) are both included in the capacitive sensor described in [7]. This capacitive sensor uses a MEMS capacitor that acts as a microphone, and the changes in capacitance are measured, amplified, and sent to the programmable filter bank. The programmable filter bank consists of two stages of differential capacitively coupled current conveyers (C^4s) with a buffer between the two stages. The biases of the C^4s are set very accurately by programming floating-gate transistors. The programmable filter bank is programmed to behave similarly to the human cochlea. Therefore, the center frequencies follow an exponential spacing, and the bandwiths are very narrow and have a moderate amount of resonance. Using the floating-gate transistors, the effects of device mismatch can be programmed out.

while simultaneously allowing a large output signal (up to $1V_{rms}$), meaning that this capacitive sensor is more that sufficient to generate desirable signals for use by the following filter bank and should interface with the filter bank very well. Additionally, this capacitive sensor operates at an SNR of 78dB while consuming only $1\mu W$ of power. Thus, the overall system, including the filter bank, consumes very little power and could be used in embedded applications such as hearing aids.

The resulting analog signal representing the acoustical waveform is then passed to a bank of programmable analog filters that act to decompose the signal into the individual frequency components. Using the human ear as inspiration for this process, since the ear efficiently decomposes sounds into the individual components, the bank of bandpass filters are programmed using floating-gate transistors to have exponentially spaced center frequencies. Also, having a moderate amount of resonance ($Q \approx 30$ as in biological systems) is also desirable for better isolation of the center frequency. Additionally, the individual bandpass filters must be compact since they are to be placed in a large array. Figure 4 shows the compact bandpass filter element that we have developed for this application. The remaining chapters of this thesis discuss how we have gone about designing and building this type of programmable bandpass array that will operate as a smart interface for an audio sensor.

1.5 Applications for low-power audio systems

Being able to incorporate extremely low-power audio sensors into portable electronics opens a wide range of signal-processing opportunities. With this new system we have developed, wherever an audio sensor is placed, additional analog circuitry can be added to the sensor to perform a large amount of signal processing at the sensor while not significantly altering the amount of power consumed by the sensor. Not only can frequency decomposition be performed at the sensor level, but by adding additional analog circuitry, noise suppression can be performed to reduce background noise [8],

or a variety of speech recognition algorithms can be used [3, 9, 10]. Additionally, high-quality hearing aids can be developed in which the batteries would seldom need to be replaced, and additional functionality can be added to cochlear implants where power constraints are a major concern.

Being able to use low-power, smart sensors could revolutionize the way consumer electronics are built. The following chapters discuss how we have proceeded in this endeavor.

CHAPTER 2

BIOLOGICAL INSPIRATION

Previous attempts at creating low-power analog audio frequency decomposition blocks have been constructed, and they have used the biology of the human ear [11, 12, 13, 14, 15, 16], just as our programmable filter bank will, too. Looking to biology for inspiration, both in the past and in the present, is done for two major reasons. First, the biological ear efficiently decomposes all incident sounds into the respective frequency components using very little power and also performing in real time. Therefore, biology serves as a good example for how this frequency decomposition can be done. Second, the human ear is often the end user of an audio processing system. Therefore, listening to an audio system that works like the biological ear will be more aesthetically pleasing than listening to a system based upon another concept. Since these early "silicon cochleae" model the human ear, as does the programmable filter bank presented here, the following is a brief description of the operation of the biological audio front end, which is the human cochlea.

2.1 Cochlea Biology

The cochlea is a small, fluid-filled bone that comprises the inner ear, as shown in Figure 5. The major function of the cochlea is to transduce acoustical waveforms into electrical signals that are meaningful to the higher centers of the brain. The cochlea does so by decomposing incident sound waves into the individual frequency components since most higher-brain centers operate on a tonotopic map, meaning that specific areas deal primarily with specific frequencies. Simply put, the cochlea, is the front end of the biological auditory system.

Sound waveforms reach the cochlea by entering through the outer ear, but the signals remain relatively unscathed until they contact the ear drum, which is the

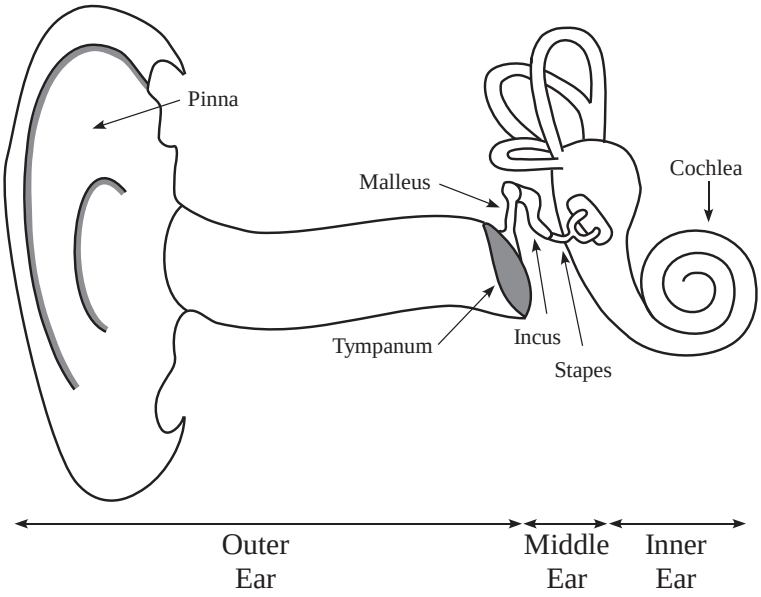


Figure 5. A cut-away view of the human ear showing the three sections of the ear. The outer ear includes the pinna, the ear canal, and the the tympanum (ear drum). The middle ear is composed of three small bones, or ossicles, that work together for gain control and for impedance matching between the outer ear and the inner ear. The inner ear (cochlea) is the snail-shaped bone in which the incoming sounds are decomposed into the respective frequency components.

pathway to the middle ear. The middle ear is composed of three bones (the malleus, incus, and stapes) that work to bring about impedance matching between the outside world and the cochlea and also contribute gain control so that very loud sounds will not damage the ear.

The middle ear connects to the cochlea (inner ear) through the oval window, which is an opening to one of three fluid-filled chambers within the cochlea, as shown in the cross section of Figure 6. The scala vestibuli, to which the oval window is connected, and the scala tympani are connected at the apex (far end of the cochlea), while the scala media is isolated from the other chambers. Flexible membranes called the Reissner's membrane and the basilar membrane separate the three chambers from each other. The organ of Corti is attached to the basilar membrane inside the scala media, as shown in Figure 6. Contained within the organ of Corti are hair cells that relay signals to the eighth cranial nerve whenever the basilar membrane undergoes

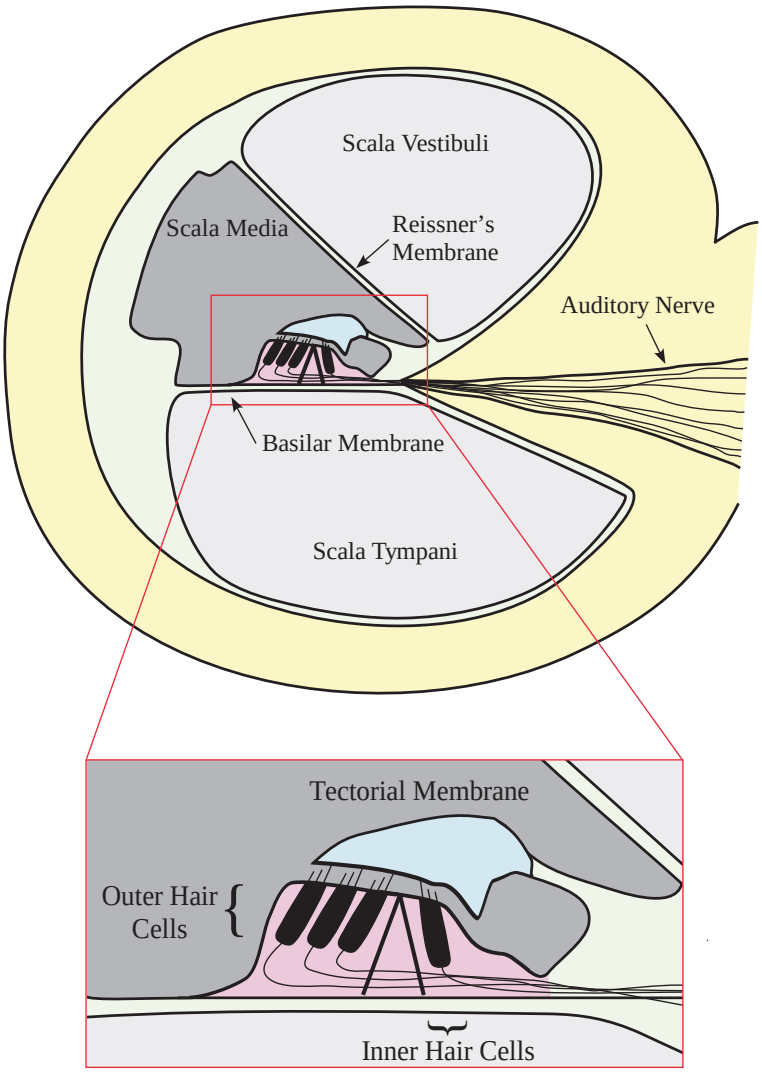


Figure 6. A cross section of the human cochlea with a close up of the organ of Corti. Within the bone of the cochlea are three fluid-filled chambers that are separated by two membranes. The input to the cochlea is in the scala vestibuli, which is connected at the apical end to the scala tympani. Pressure differences between these two chambers leads to movement in the basilar membrane. The scala media is isolated from the other two chambers. The zoomed-in portion of the figure shows the organ of Corti in greater detail. Four rows of hair cells extend from the lower portion of the organ of Corti. As the basilar membrane resonates, the lower portion of the organ of Corti pivots around a different axis than does the tectorial membrane, thus causing the stereocila to bend back and forth. This motion of the stereocilia modulates the amount of positive ions (mostly K^+) that flows into the inner hair cells (IHCs) and, hence, the membrane potential of the IHCs. The three rows of outer hair cells are primarily involved in gain control and mostly receive efferent input from higher centers of the brain.

motion [17, 18].

The shape of the basilar membrane varies systemmatically through the length of the cochlea. At the basal end, which is the front end, the basilar membrane is very narrow ($\approx 0.04mm$), but it gets wider towards the apical end ($\approx 0.5mm$) [19]. Also, the basilar membrane is more taut at the front end and looser at the rear. Further, the thickness, or heaviness, of the basilar membrane also changes down its length. As a result, the basilar membrane resonates at different frequencies along its length, much in the same way as would a continuoum of masses of differing weights suspended by springs of varying spring constants, as depicted in Figure 7a. In essence, this view can be used to model the basilar membrane, and each position along its length is governed by a second-order equation. Each location, therefore, resonates at a specific frequency and, thus, acts as a bandpass filter, as is shown in Figure 7c. These "bandpass filters" in the basilar membrane follow an exponential spacing in center frequency for a linear spacing along the length of the basilar membrane, as the tonotopic map of Figure 7b illustrates. Since the basilar memebrane plays an integral roll in the transduction of sound into electrical signals, it has become the center of attention when modeling the cochlea.

If a pure sinusoidal waveform is the input to the cochlea, the basilar membrane will resonate strongly at one point and will quickly die out in both directions. However, there is asymmetry in the attenuation on either side, which means that a bandpass filter mirroring this function should have a sharper roll off in the high frequencies than in the low frequencies. Also of interest is that the other portions of the basilar membrane that are in close proximity to the point of resonance oscillate out of phase with the position of resonance. Viewing the basilar membrane for a constant sinusoidal input over time lends to the appearance that the waveform travels along the length of the basilar membrane. However, this is in error; the fluid in the cochlea is an incompressible fluid, and the pressure induced at the opening to the cochlea

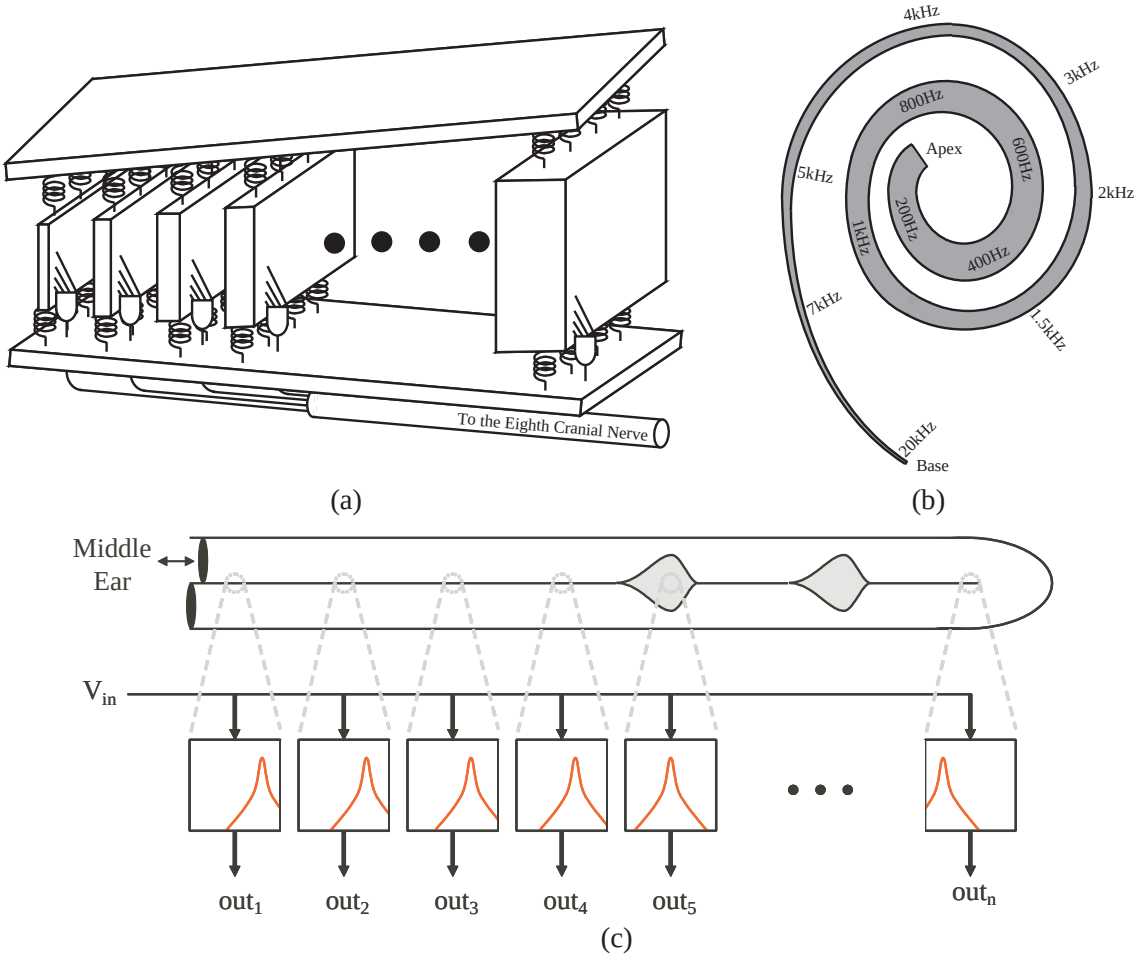


Figure 7. Illustration of the resonance properties of the cochlea. (a) The basilar membrane can be thought of as a set of blocks on springs. Each block is of a different mass (slightly larger than the one preceding it), so each block resonates at a different frequency with second-order dynamics. Higher frequencies resonate smaller blocks (near the front) while lower frequencies resonate larger blocks (near the back). Hair cells sense the resonance and send signals down the eighth cranial nerve. Coupling occurs through the fluid. (b) Tonotopic map of the human cochlea showing that the basilar membrane has exponential changes in the resonance frequency for linear distances down the length of the cochlea. (c) Bandpass view of the basilar membrane. Viewing a "rolled-out" version of the basilar membrane, which is the most significant portion of the cochlea for cochlear modeling, the cochlea can be described as a bank of bandpass filters in parallel. Each bandpass filter represents a singular location on the basilar membrane. These representative bandpass filters follow an exponential spacing and have a moderate amount of resonance.

influences the entire basilar membrane at once.

For more complicated sound waveforms, such as speech, the resonating response is repeated at each location where there is power in the spectrum of the sound. As a result, the cochlea works to perform a spectral decomposition on any incoming waveform.

2.2 Previous Silicon Cochlear Models

Early versions of silicon cochlear models, starting with the work of Lyon and Mead [20], did not use an array of bandpass filters in parallel as would be expected from biology, but instead attained pseudo-bandpass responses by cascading lowpass filters, as is shown in Figure 8a. These lowpass filters were based on a class of circuits dubbed second-order sections (SOSs) because they have a second-order lowpass transfer function. Figure 8b shows the major G_m-C filter topology that was used [1]. This circuit has the simple property that a single bias value sets the corner frequency and another single bias value sets the amount of resonance of the filter.

To model a human cochlea with these lowpass filters, SOSs were arranged in a cascade with the output of one serving as the input to the next stage. Resistor lines that were formed by using the inherent small resistance of polysilicon, which is used for gates in CMOS processes, were placed along the length of the cascade, and bias voltages were placed on both ends. Therefore, the tapped resistor lines acted as large resistive dividers and yielded linearly spaced bias voltages. Since each bias voltage was connected to the gate of a MOSFET and because of the exponential nature of MOSFETs running in the subthreshold regime, linearly spaced bias voltages translated into exponentially spaced currents. As a result, the corner frequencies of each element changed exponentially.

The overall cochlea model, thus, took the form of a cascade of lowpass filters where each filter had a lower corner frequency than its predecessor. Any input signal would

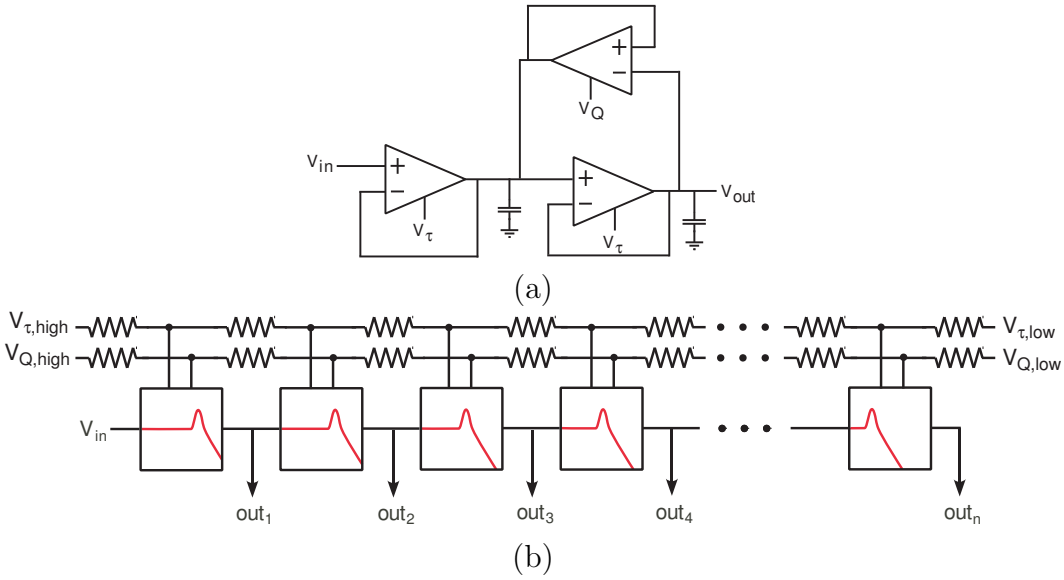


Figure 8. (a) Lowpass filter often used as the core filter in cochlear models. (b) Generalized schematic of previous silicon models of the human cochlea. These previous models consisted of a cascade of second-order, G_m-C, lowpass filters. These filters were biased by using resistive lines acting as large resistive dividers. Equal spacing along the resistors provides linear spacing of the bias voltages, and since these resistors were biasing the gates of transistors running in the subthreshold regime, the currents flowing through the bias transistors had exponentially spaced currents.

enter the cascade and travel down the line unimpeded until it reached a point where it was emphasized (at the resonance point). Only signals of lower frequencies would be passed along to the next stage [1].

Since the convolution of two transfer functions in the frequency domain is equivalent to a multiplication, the output of stage n is a multiplication of its own transfer function and that of all those preceeding it. Therefore, the high frequency roll offs of each stage are much steeper than the stage's own contribution of -40dB/dec. Also, the resonance, or Q peak, for each stage is multiplied by those preceeding it (which are offset to slightly higher frequencies), and the the stage's resonance point is increased in magnitude and also broadened. In essence, the filtering properties of a particular stage take on properties of a bandpass filter. Also, adding a differentiator to the circuit also helps the cascade model achieve a pseudo-bandpass response [12].

This cascade approach had two major sources of problems. The first was in the

cascade itself. By cascading many (often more than 100) of these lowpass filters, the cascade was prone to accumulating noise, phase, and delay. For example, any noise in the first stage would propagate on to the next stage and combine with the noise in that stage. This combined noise would then propagate down the line and combine with the noise of each individual stage, thus making the overall noise very large. Also, if any stage ceased working (due to a blown oxide, latch up, *etc.*), then all subsequent stages at lower frequencies would no longer be able to perform their duties. (We are fortunate that biology does not operate on this same principle since hearing loss typically occurs at the high frequencies first.) The second major source of problems was with the imprecision of setting the corner frequencies due to mismatch within the devices. These problems were addressed [12], but the solutions require large amounts of real estate. Additional improvements to this cascade approach have been made by several researchers [11, 12, 13, 14], but we believe that vast improvements to this signal processing block can be made by using bandpass filters instead.

2.3 A Bandpass Alternative

Since the basilar membrane essentially functions as an array of bandpass filters in parallel and since the previous cochlear models have so many problems associated with its cascade structure, we propose building an array of bandpass filters for improved audio performance. Like biology, these bandpass filters should be exponentially spaced [17] and have large amounts of resonance in each stage ($Q \approx 32$) [21]. Adding this resonance is for the dual purposes of modeling biology more closely and simply giving better isolation of the center frequency. Either first- or second-order slopes outside the passband may be used to model the basilar membrane's displacement or velocity, respectively [22]. Additionally, floating-gate transistors will be used to solve the problems associated with mismatch, as will be discussed later.

This use of active bandpass filters to model the human cochlea is a marked change

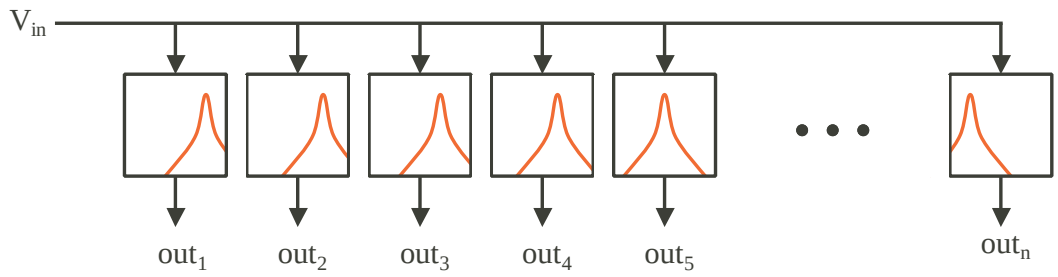


Figure 9. A model of the cochlea that emphasizes the resonance nature of the cochlea. Instead of using a cascade of lowpass filters, this model uses a bank of bandpass filters with second-order slopes.

in the method in which silicon cochlear models are done. Prior to this work, all active silicon cochlear models have used a cascade of lowpass filters. However, since the beginning of this work in 2001 and the resulting first publication in 2002 [23], several of the researchers that were already working on silicon cochlear models have switched paradigms and have begun using a bandpass approach [16, 15, 24], and the new researchers who have entered the field are using bandpass filters [25, 26, 27]. In fact, since our work was first published, virtually all of the silicon cochlear models have used an array of bandpass filters, just as our work introduced the concept.

CHAPTER 3

THE CAPACITIVELY COUPLED CURRENT CONVEYOR

The most fundamental block required to build an auditory front end based on the biology of the human cochlea is a compact, low-power, continuous-time, bandpass filter. While first-order slopes are sufficient to model the displacement of the basilar membrane, second-order slopes are better since they model the velocity of the basilar membrane [22]. Also, these filters should have large resonance ($Q \approx 32$) to help mimic the cochlear response and to isolate specific frequencies [21].

In this chapter, we will discuss a new compact bandpass-filter element that can be used to meet the specifications needed in cochlear modeling and can also be used in a wide variety of signal-processing applications. This discussion will first focus on the basic principles of the bandpass-filter element specifically geared towards cochlear modeling. Then, the discussion will delve into a general analysis of this bandpass filter and will provide a synthesis routine for designing bandpass filters to meet any given specifications.

3.1 Overview of the C^4 for Cochlear Modeling

The capacitively coupled current conveyor (C^4) is a bandpass filter that has been designed for a variety of filtering applications. Due to its compact size and ease of tuning, the C^4 is the primary filter used in our cochlear model. This initial discussion of the C^4 focuses on the properties of the C^4 that make it so well suited for cochlear modeling. As a result, the discussion of this section focuses on the properties of the C^4 when biased in the subthreshold regime with a narrow bandwidth. Section 3.2 provides a more generalized description of the C^4 and its uses in other types of filtering applications.

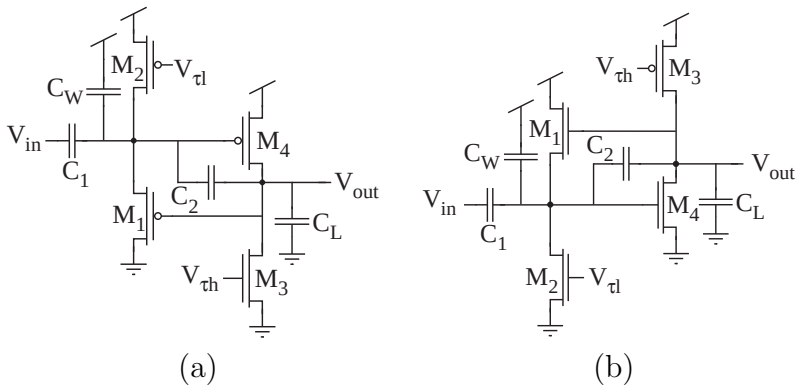


Figure 10. Schematic of the capacitively coupled current conveyor (C^4) (a) pFET-based C^4. (b) nFET-based C^4. The C^4 is the fundamental bandpass-filter element used in our cochlear model.

3.1.1 The C^4 Element

The capacitively coupled current conveyor (C^4) has been used as a starting point for developing the required filter for the overall bandpass array. The C^4 was previously introduced [28] and used in a system application [29]. However, the initial theory on the C^4 was developed for separated corner frequencies, as was its model system, the autozeroing floating-gate amplifier [30]. However, certain properties of the C^4 were not properly appreciated but play a significant role in creating higher-order filters and systems with cochlea-like responses. These additional qualities were further characterized [31, 32] and are summarized here.

The C^4 is the capacitively based bandpass filter shown in Figure 10. The C^4's corner frequencies are electronically tunable and can be set independently of one another. The frequency response of the C^4 is governed by

$$\frac{V_{out}}{V_{in}} = -\frac{C_1}{C_2}\frac{s\tau_l(1-s\tau_f)}{s^2\tau_h\tau_l + s(\tau_l + \tau_f(\frac{C_o}{\kappa C_2}-1)) + 1} \tag{1}$$

where the time constants are given by

$$\tau_l = \frac{C_2 U_T}{\kappa I_{\tau_l}} \qquad \tau_f = \frac{C_2 U_T}{\kappa I_{\tau_h}} \qquad \tau_h = \frac{C_T C_O - C_2^2}{C_2}\frac{U_T}{\kappa I_{\tau_h}} \tag{2}$$

and where the total capacitance, C_T, and the output capacitance, C_O, are defined as $C_T = C_1 + C_2 + C_W$ and $C_O = C_2 + C_L$. The currents I_{τ_l} and I_{τ_h} are the currents

through M_2 and M_3, respectively in Figure 10b. With normal usage, τ_f is very small, and the zero it produces lies far outside of the operating range. The plots of Figure 11 show data from a C^4 fabricated in a 0.5μm process available through MOSIS that summarizes the response of the C^4. The C^4's transfer function and other significant properties are analytically computed in Appendix A.

The C^4 has the properties of a bandpass filter with first-order slopes and a bandpass gain set by the ratio of the two coupling capacitors as $A_v \approx -C_1/C_2$. The overall time constant of the filter, which gives the center frequency, is

$$\tau = \sqrt{\tau_l \tau_h}. \tag{3}$$

By tuning the filter such that $\tau_h > \tau_l$, resonance occurs, and the quality factor, Q, is

$$Q = \sqrt{\frac{\tau_h}{\tau_l}} \frac{1}{1 + \frac{I_{\tau_l}}{I_{\tau_h}}(\frac{C_O}{\kappa C_2} - 1)}. \tag{4}$$

Figure 12 is a plot of the quality factor versus the ratio of I_{τ_l}/I_{τ_h}. The maximum value occurs when I_{τ_h} is slightly larger than I_{τ_l} for the given capacitor sizes and is given by

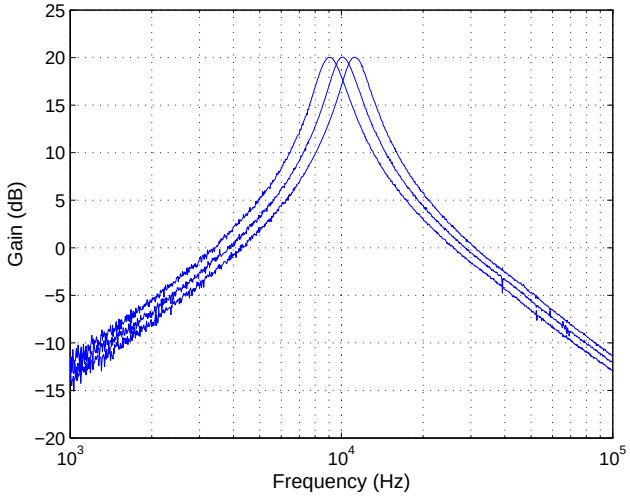
$$Q_{max} = \frac{1}{2}\sqrt{\frac{\kappa\,(C_T C_O - C_2^2)}{C_2\,(C_O - \kappa C_2)}}. \tag{5}$$

By removing the drawn C_2 capacitor, a high-gain C^4 is created since the midband gain is $A_v \approx -C_1/C_2$. The overlap capacitance of the MOSFET causes there to be a small effective C_2 capacitance, so the gain is not infinite. By reducing the value of C_2, more resonance occurs. If C_2 is made sufficiently small, then the Q_{max} equation can be reduced to

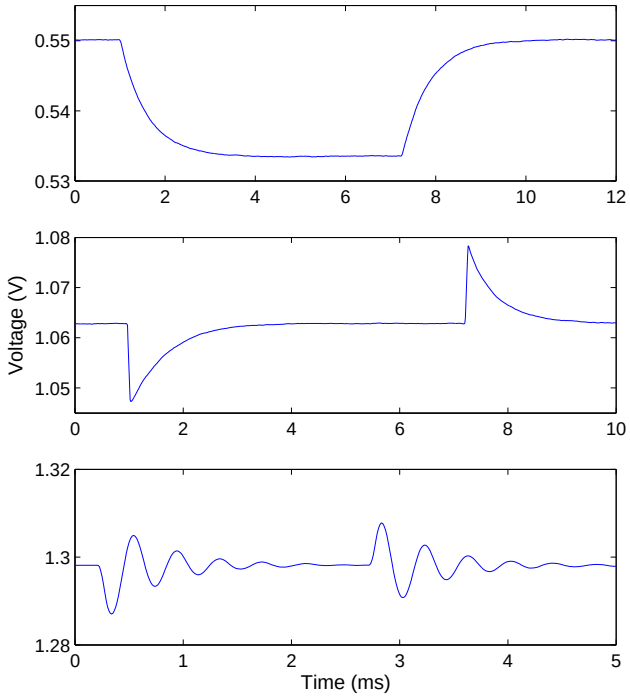
$$Q_{max} \approx \frac{1}{2}\sqrt{\frac{\kappa\,(C_1 + C_W)}{C_2}}. \tag{6}$$

3.1.2 Input-Capacitance Shifts

One potentially hazardous trait of the C^4 is that the input capacitance of the circuit does not necessarily remain constant, but it varies based on frequency. This can lead



(a)



(b)

Figure 11. Experimental measurements from a C^4 illustrating various modes of operation. (a) Frequency response of the C^4 showing fine tuning of the bandpass response. (b) Step response of the C^4. (Top) Step response of the C^4 when biased as an integrator. (Middle) Step response of the C^4 when biased as a differentiator. (Bottom) Step response of the C^4 when the two corner frequencies have crossed each other.

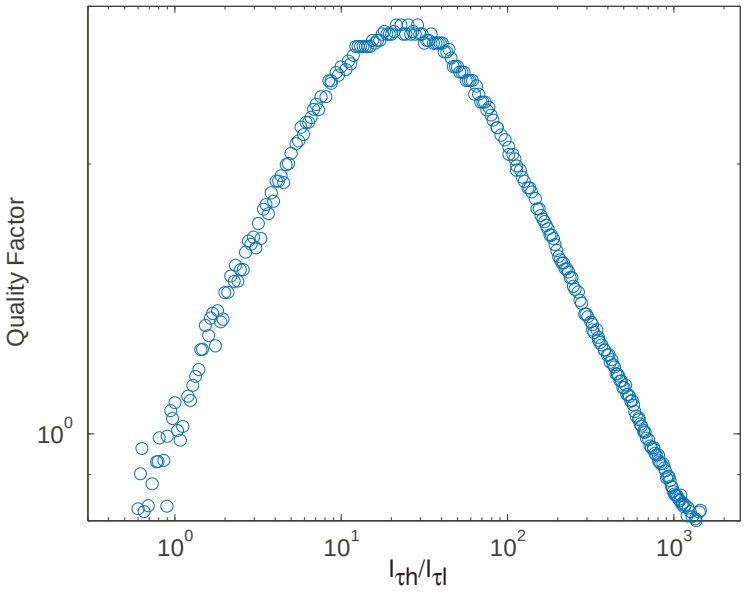


Figure 12. Quality factor versus the ratio of I_{τ_h}/I_{τ_l}. The maximum value of the quality factor, Q, is set by a ratio of capacitances.

to problems in creating higher-order filters by cascading C^{4}s if not properly taken into account.

The input capacitance can easily be found for the cases of very-low frequencies and very-high frequencies. For very-low frequencies, the middle node is effectively an AC ground because of the high-gain amplifier; hence, the input capacitance is simply $C_{in} = C_1$. For very-high frequencies, the transistors can no longer follow the signals, so the C^4 reduces to a network of capacitors and the input capacitance becomes the series-parallel combination of the capacitances in this network. The input capacitance for the two extreme cases are given by

$$C_{in}(f \to 0) = C_1$$
$$C_{in}(f \to \infty) = C_1 \| \left(C_W + C_2 \| C_L\right) \approx C_1 \| C_W \tag{7}$$

The approximation holds in the case when C_2 is significantly smaller than the load capacitance, C_L. Figure 13 shows results of a SPICE simulation in which the input capacitance was computed. This figure shows that the transition region between these

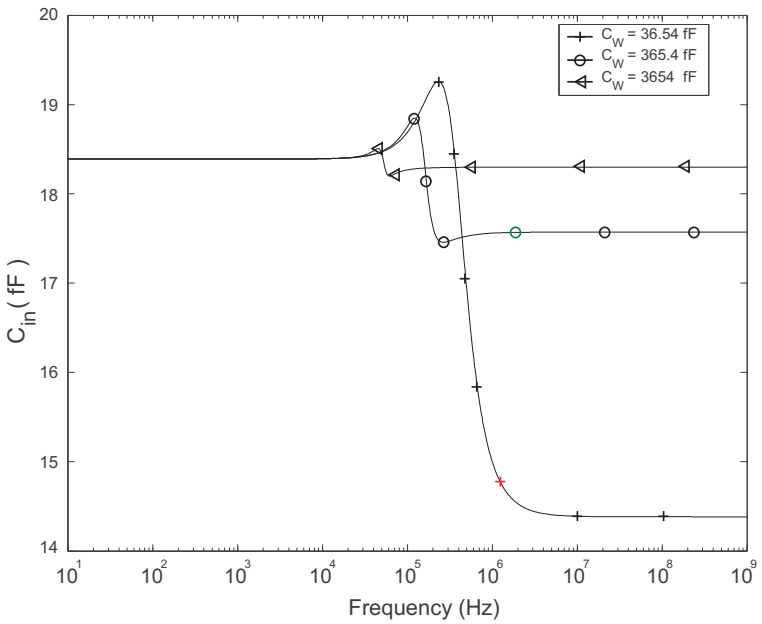


Figure 13. Changing the input-capacitance shift by varying C_W. C_W was increased by factors of 10. As C_W became larger, the high-frequency input capacitance approached the low-frequency value.

extremes of C_{in} is in a fairly confined frequency band near the center frequency of a C^4 with a tightly tuned bandwidth.

Increasing the drawn size of C_W is a good choice for reducing the shifting input capacitance for the simple reason that the larger the value of C_W, the more closely $C_1 \| C_W \approx C_1$, and hence the more closely the high-frequency C_{in} approaches the the low-frequency C_{in}. Figure 13 shows the results of 10-fold increases in the capacitance of C_W. The larger that C_W is drawn, the less the effect of the input capacitance shift on the system. Another way of reducing the effects of the shifting input capacitance on other circuits is to place a buffer in front of the C^4 so that the previous circuit always sees the same load capacitance.

3.1.3 Cascaded C^4s

When building a filter to model cochlear dynamics, desirable characteristic are high resonance ($Q \approx 32$) and sharp slopes in the stopband. These filters can be made by using the C^4 as a basic building block. Since a quality factor of $Q \approx$ 5-6 is easy

to design by using a C^4 without a drawn feedback capacitor, C_2, a cascade of two such elements brings the overall effective quality factor, Q_{eff}, into the desired range. Therefore, a C^4 second-order section (C^4 SOS) is shown in Figure 14a as a cascade of two C^4s with a buffer between the stages. Here the "second-order" refers to the high-frequency roll offs and not the order of the transfer function. Therefore, this filter is comparable to the SOSs of the cascade cochlear models [20, 33, 12, 14].

The plot of Figure 14b shows frequency responses from single C^4s and a cascade of two C^4s fabricated in a 0.5μm process available through MOSIS [34]. SPICE simulations show how closely the model fits the data. These data show that cascading two C^4s greatly increases the amount of resonance in the overall filter. We have achieved $Q_{eff} = 70$ at 1MHz for a cascade of two C^4s [34, 35].

By being able to cascade two C^4s, the desired response in terms of resonance and stopband roll-offs is achieved for cochlear modeling expectations. Additionally, the C^4 can be used to attain a wide bandwith for a bandpass response, and it can be used as a basic filter element in designing higher-order filters. The following section, therefore, explores more of the details of the C^4 in these terms.

3.2 General Analysis of the C^4

We will now proceed to a more careful study of the operation of the C^4, including wide-band operation, and will show that this filter is useful for a large range of filtering applications beyond cochlear modeling [36]. This analysis will focus on the properties of a single-ended C^4; however, the C^4 can be made differential by placing two C^4s in parallel (one for the positive input and one for the negative input) and optionally adding a common-mode feedback (CMFB) stage using a standard differential amplifier with high loop gain [37]. By adding programmability through floating-gate transistors, as will be discussed in Chapter 5, the complete C^4 is as shown in Figure 15.

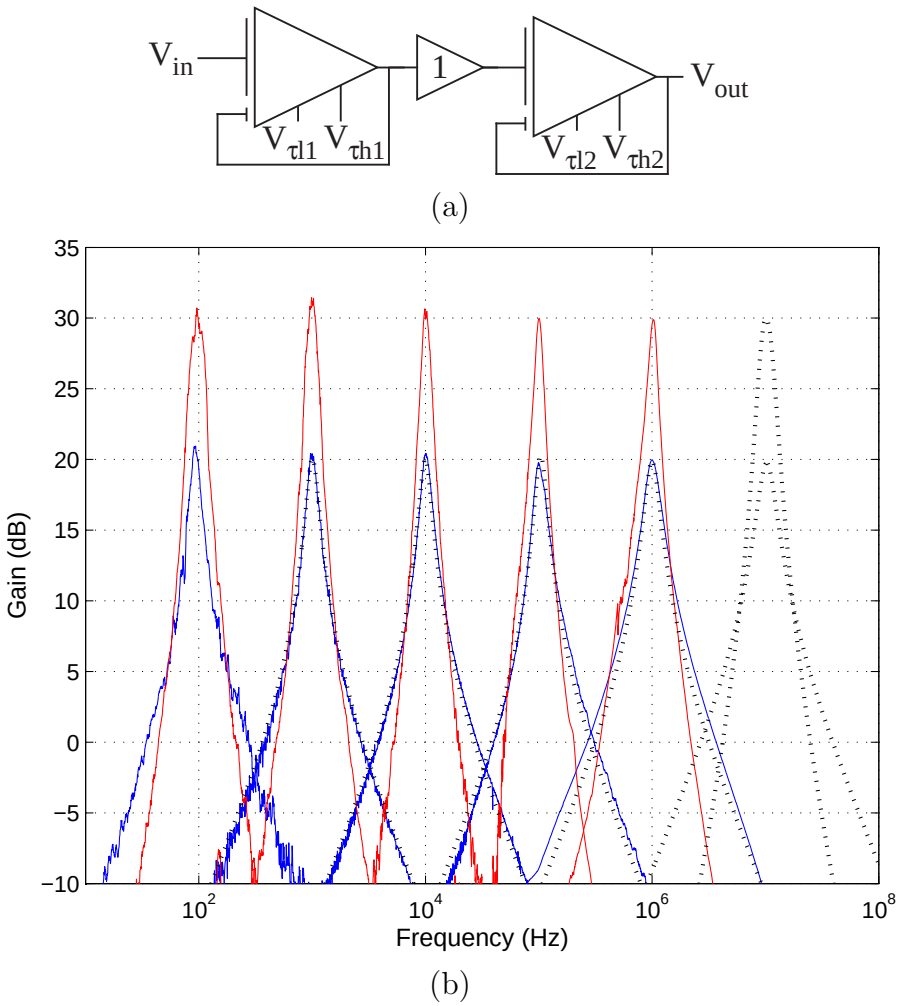


Figure 14. A C^4 second-order section (C^4 SOS). (a) Shorthand notation of the C^4 SOS, where each amplifier symbol represents a single C^4, and the "extra lines" on the inputs represent the capacitive inputs. The asymmetric sizes of the capacitive-input lines represent the greatly reduced size of the feedback capacitor in the C^4. By using a cascade of two C^4s, the overall output has second-order slopes, and the Q peaks of each stage are multiplied for the output of the cascade. A buffer is used between the two stages to isolate the two stages and to give the first stage a steady load capacitance. (b) Frequency responses from a cascade of two C^4s. The red trace shows the output of the complete filter, while the blue trace shows the output of only the first stage. Simulation results with the black dashed lines show that simulation matches closely to the actual performance. The output buffer ceased functioning properly at approximately 5MHz, but simulation results show that the C^4 continues to operate above this frequency.

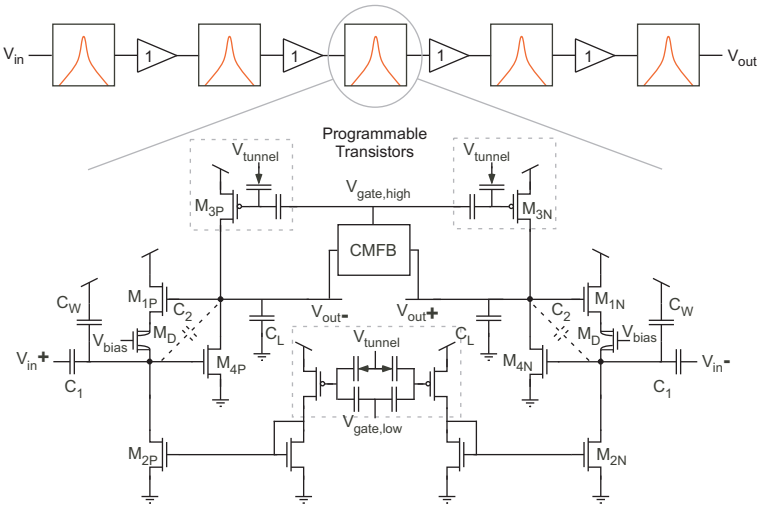


Figure 15. Schematic of the complete C^4. The C^4 can be made differential by placing two single-ended C^{4}s in parallel where each is for either the positive or negative inputs. Additionally, common-mode feedback (CMFB) can be added by simply using a high-gain differential amplifier. Programmability of the corner frequencies can be achieved by using floating-gate transistors as current sources to set each time constant. The two M_D transistors are short channel-length devices used for increasing the output linear range. The C^4 can also be used as a simple filter element when cascading several C^{4}s to create a high-order bandpass filter.

Figure 16a repeats the schematic of the single-ended C^4 and includes an additional transistor, M_D, which is used for increasing the output linear range of the C^4. M_D is a device with a short channel length that provides source degeneration to the feedback stage. Also, Figure 16b shows the origin of the C^4 which is the autozeroing floating-gate amplifier (AFGA) [38, 39, 30] from which the C^4 was derived [28]. The AFGA typically had separated corner frequencies, and the inital generalized analysis of the C^4 will also focus on the case of separated corner frequencies.

Figure 16c shows the reduced circuit illustrating the high-frequency behavior of the C^4, and Figure 16e shows the resulting small-signal circuit. From this circuit, the resulting transfer function is

$$\frac{V_{out}}{V_{in}} = -\frac{C_1}{C_2}\frac{1 - s\tau_f}{1 + s\tau_h} \tag{8}$$

where the time constants are given by

$$\tau_h = \frac{(C_T C_O - C_2^2)}{C_2 g_{m4}} \qquad \tau_f = \frac{C_2}{g_{m4}} \tag{9}$$

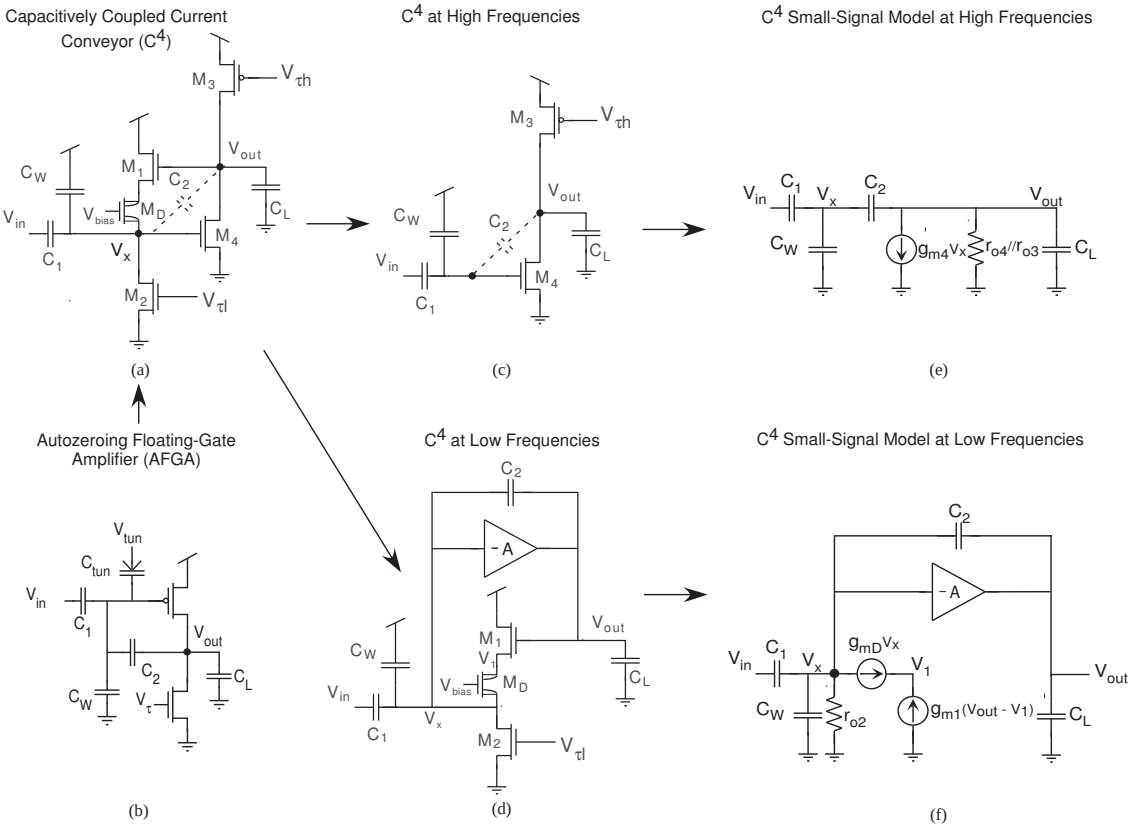


Figure 16. Qualitative description of the C^4. (a) C^4 schematic. The time constants are set by the current-source transistors M_3 (high-frequency corner) and M_2 (low-frequency corner). (b) The C^4 approach has its roots in the autozeroing floating-gate amplifier (AFGA) circuit [30]. The upper time constant is set by the current-source transistor, and the lower time constant is set by the balance of electron tunneling and hot-electron injection. (c) Equivalent circuit schematic of the C^4 at high frequencies in which the feedback loop has minimal effect on the circuit response. (d) Equivalent circuit schematic of the C^4 at low frequencies in which the common-source amplifier with transistor M_4 acts as a constant gain amplifier with gain A. (e) Small-signal model for the high-frequency equivalent circuit. (f) Small-signal model for the low-frequency equivalent circuit.

and where the total capacitance and the output capacitance are given by $C_T = C_1 + C_2 + C_W$ and $C_O = C_2 + C_L$, respectively. The passband gain of the filter is set by capacitor ratios as $A_v = -C_1/C_2$. The zero in this expression, determined by τ_f, is due to capacitive feedthrough from the input to the output of the amplifier, or the effective circuit when operating at a sufficiently high frequency such that the amplifier behavior on the output voltage is negligible. The capacitive feedthrough normally has little effect on the bandpass filter operation in either frequency or amplitude; the resulting feedthrough gain and τ_f should be calculated to verify this assumption when designing the C^4 bandpass filter.

From the simplified circuit of Figure 16c and e, we can estimate the noise and the signal-to-noise ratio (SNR) for this wideband amplifier. The output thermal-noise voltage integrated over the entire bandwidth of interest (set by τ_h) is computed as

$$V_{noise} = \sqrt{q\left(\frac{I_4}{g_{m4}}\right)\frac{C_T}{C_O C_2}}, \tag{10}$$

where $q = 1.6 \times 10^{-19}$C, and I_4 is the bias current flowing through M_4. For the wideband case for the complete C^4, the noise is divided by a term that is typically close to unity and is given by $1 + g_{m1}(C_O/C_2 - 1)/(\kappa g_{m4})$. For subthreshold-current levels, the noise takes on the form of kT/C noise where the effective capacitance is $\kappa C_2(C_O/C_T)$. The output-referred linear range is given by $U_T C_T/(\kappa C_2)$ (subthreshold operation) and $V_{on}C_T/C_2$ (above-threshold operation), assuming $C_T C_O >> C_2^2$ and that $V_{on} = \kappa(V_g - V_T) - V_s$ is the overdrive voltage at the bias condition. The linearity is set by choosing the desired capacitor value for C_W, which results from the capacitive attenuation at the input. Distortion for a differential system is less than -40dB at all points (third harmonic limited) over all frequencies (largest at one-third the center frequency). The resulting SNR for this amplifier is

$$SNR = 10\log_{10}\left(\frac{1}{q}\left(\frac{I_4}{g_{m4}}\right)\frac{C_T C_O}{C_2}\right). \tag{11}$$

The SNR is directly increased by the product of $C_T C_O$ divided by C_2, resulting in

significantly smaller capacitor sizes for a given SNR than can be achieved by using other G_m-C techniques. When designing a C^4, the $1/f$ noise corner frequency should be determined for the given biasing conditions; if the $1/f$ corner is not in the passband, the the effect of $1/f$ noise can be neglected.

Figure 16d shows the reduced circuit to illustrate the low-frequency behavior of the C^4, and Figure 16f shows the resulting small-signal circuit. The primary assumption is that the amplifier between V_X and the V_{out} has a constant gain, A, because transistors M_3 and M_4 form a high-gain inverting amplifier that yields a constant gain over the frequency range of interest. Assuming that $A \gg 1$, the resulting transfer function is given by

$$\frac{V_{out}}{V_{in}} = -\frac{C_1}{C_2}\frac{s\tau_l}{s\tau_l + 1} \tag{12}$$

where

$$\tau_l = \frac{C_2}{g_{m1}}. \tag{13}$$

C_2 includes the overlap capacitance of M_4 and also the capacitance from the gate of M_1 to the source of M_D, which is small since M_D cascodes M_1. M_D is a short-channel device ($0.5\text{V} < V_A < 10\text{V}$) that is used to increase the linearity from V_{out} back to V_X. This linear range, which is given by $V_{L1} = I_1/g_{m1}$, typically falls between 0.5V and 10V.

The low- and high-frequency time contants can be set independently of each other by tuning g_{m1} and g_{m4}, respectively, which is done by tuning the bias currents flowing through transistors M_1 and M_4. The transfer function incorporating both the low- and high-frequency responses is given by

$$\frac{V_{out}}{V_{in}} = -\frac{C_1}{C_2}\frac{s\tau_l\left(1 - s\tau_f\right)}{1 + s\left(\tau_l + \tau_f(\frac{C_O}{C_2} - 1)\right) + s^2\tau_h\tau_l}. \tag{14}$$

Figure 17 shows the frequency response of the C^4 illustrating that both high (10kHz, 11kHz, 12kHz) and low (100Hz, 200Hz, 300Hz) corners can be individually tuned to the desired frequencies accurately. As was previously shown in Figure 11b, the C^4

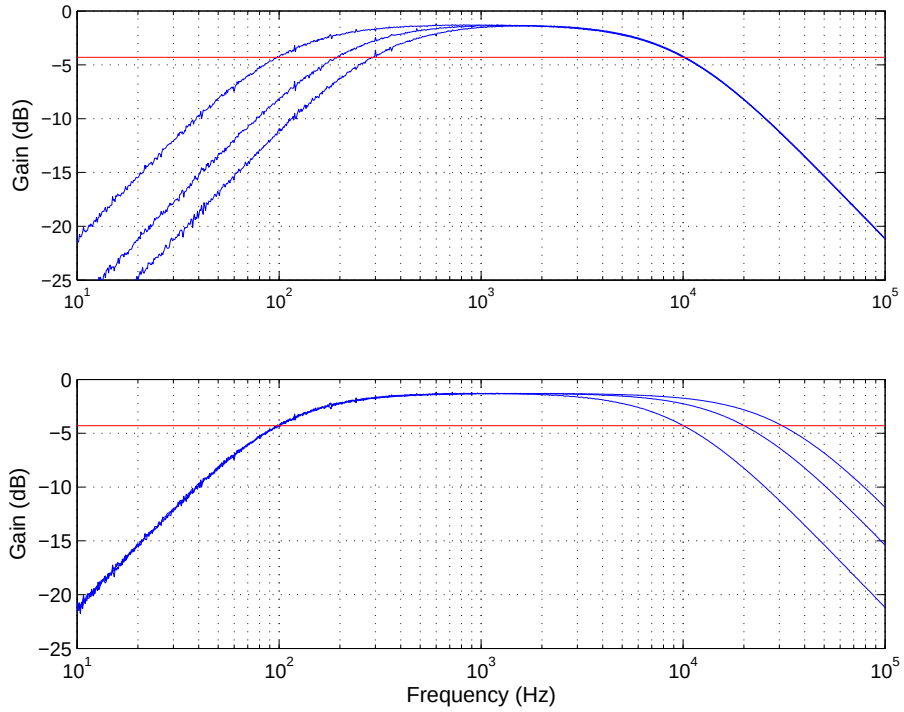


Figure 17. Frequency response of the C^4 for widely tuned corner frequencies. These measurements show that the tuning of the high- and low-corner frequencies are independent of each other.

can be biased to give a low-pass response (high-frequency approximation), a high-pass response (low-frequency approximation), or a combination of the responses leading to resonance.

By moving the time constants closer to each other, the C^4 takes on a bandpass response. Crossing the time constants introduces resonance into the filter response, as was shown in Figure 11. The resulting transfer function is

$$\frac{V_{out}}{V_{in}} = -\frac{C_1}{C_2}\frac{sC_2/g_{m1}}{1 + s\left(\frac{C_2}{g_{m1}} + \frac{C_o}{g_{m4}}\right) + s^2\frac{C_oC_T}{g_{m4}g_{m1}}} \tag{15}$$

where the capacitive feedthrough term, τ_f, is assumed to have a negligible effect on the transfer function of interest. The small-signal model shown in Figure 18 gives another method for obtaining the above results; because of the Miller effect and frequency-dependent amplifier gain, the circuit model displays effective inductance and conductance parameters on the V_X node. The Miller capacitance is amplified

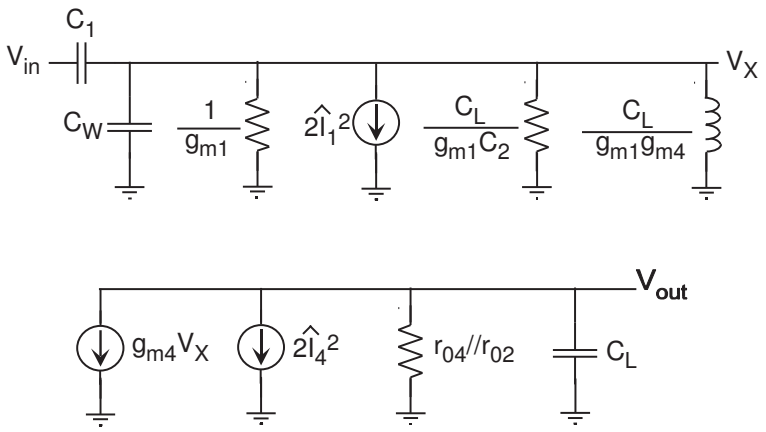


Figure 18. Small-signal model of the C^4 for $Q > 0.5$. This model shows the effective inductance and conductance that depends on real circuit parameters. This model gives intuition of the filter operation, as well as easily enabling hand calculate for linear-performance parameters for the high-Q case.

by loop gain, which is also frequency dependent; therefore, not only is this capacitance amplified, but a resulting conductance and inductance (due to the gyrator like structure) are, as well.

This circuit model can be used to compute the performance of the C^4 for $Q > 1$. The center frequency, f_{center}, is set by ($C_O C_T << C_2^2$)

$$f_{center} = \frac{1}{2\pi\tau} = \frac{\sqrt{g_{m4} g_{m1}}}{2\pi\sqrt{C_o C_T}}, \tag{16}$$

the passband gain, A_v, is set by

$$A_v = -\frac{C_1}{C_2} \frac{1}{1 + \frac{g_{m1}}{g_{m4}} \frac{C_o}{C_2}}, \tag{17}$$

and the quality factor, Q, of the resonance is given by

$$Q = \frac{\sqrt{C_T C_O}}{C_2 \sqrt{\frac{g_{m4}}{g_{m1}}} + C_O \sqrt{\frac{g_{m1}}{g_{m4}}}}. \tag{18}$$

Transistors M_1 and M_4 can operate in weak, moderate, or strong inversion depending on the desired frequency response. As can be seen from the above equations, the corner frequency and the quality factor depend on the transconductances and, therefore, the DC bias current. Thus, the filter element can be easily fine-tuned after fabrication to the desired corner frequencies and Qs by tuning the g_{m1} and g_{m4}.

For these filters, capacitor ratios set a maximum quality factor, Q_{max}. Figure 12 shows that Q changes with the ratio of I_{τ_h}/I_{τ_l}, or accordingly, g_{m4}/g_{m1}, which is consistent with (18). The plot illustrates that a maximum Q peak occurs for a certain value of I_4/I_1 (and thus g_{m4}/g_{m1}) and decreases as the ratio is either increased or decreased. Q_{max} is given by

$$Q_{max} = \frac{1}{2}\sqrt{\frac{C_T}{C_2}} \quad \text{where} \quad \frac{\mathrm{g_{m4}}}{\mathrm{g_{m1}}} = \frac{\mathrm{C_o}}{\mathrm{C_2}} \tag{19}$$

Increasing Q requires either increasing C_T or decreasing C_2; C_2 includes the gate-to-drain capacitance (overlap capacitance) of M_4 and the effective capacitance from gate to source of the M_1 and M_D transistor combination. The resulting center frequency at Q_{max} is

$$f_{center} = \frac{g_{m4}}{4\pi C_o Q_{max}} = \frac{g_{m1}}{4\pi C_2 Q_{max}}, \tag{20}$$

and the gain at the center frequency is $A_v = -C_1/2C_2$.

Next, we address the noise generated by the filter. The generated frequency dependent noise model for the C^4 amplifier is computed as

$$\hat{V}_{out}(s) = \frac{\sqrt{2}\hat{I}_4(s)g_{m1}(1 + sC_T/g_{m1}) + \sqrt{2}\hat{I}_1(s)g_{m4}(1 + s\tau_f)}{s^2 C_O C_T + s(C_L g_{m1} + g_{m4}C_2) + g_{m1}g_{m4}} \tag{21}$$

where $\hat{I}_1(f)$ and $\hat{I}_4(f)$ are the thermal-noise quantities contributed by M_1 and M_4, respectively. These thermal-noise expressions are given by

$$\hat{I}_1(f) = 2qI_1\Delta f \tag{22}$$

$$\hat{I}_4(f) = 2qI_4\Delta f \tag{23}$$

where Δf is the bandwidth of the filter and I_1 and I_4 are the bias currents. We can solve for the in-band noise by integrating over the bandwidth, or solve for the noise over the entire spectrum by integrating over all frequencies. In most cases, the in-band noise is by far the largest noise component. When integrating over the bandwidth, we center our integration around f_{center} and integrate over the bandwidth

($\Delta f = f_{center}/Q$). Solving for the total in-band noise, we get

$$\hat{V}_{out} = \int_f \frac{\sqrt{2}\hat{I}_4(f)C_T - j\sqrt{2}\hat{I}_1(f)g_{m4}\tau}{(g_{m1}C_L + g_{m4}C_2)}df \tag{24}$$

$$\hat{V}_{out} = \frac{\sqrt{qI_4\frac{C_T(C_2g_{m4}+C_Og_{m1})}{C_O}} - j\sqrt{qI_1(\frac{C_2}{g_{m1}} + \frac{C_O}{g_{m4}})g_{m4}}}{(g_{m1}C_L + g_{m4}C_2)}, \tag{25}$$

where we defined the effective noise bandwidth as $1/(4\tau Q)$ [40]. Noise at low-frequencies (not in band) is nearly constant independent of frequency as determined by the thermal noise level, and the total noise in this region is the same as for the wideband stage, which is important if adding together the results of multiple elements, as in a programmable filter [28].

We simplify the noise modeling when biased in the Q_{max} case, which helps in providing intuition about the noise behavior over the range of potential bias currents. For the Q_{max} case, the noise expression becomes

$$\hat{V}_{out} = \sqrt{\frac{I_4}{g_{m4}}\frac{qC_T}{2C_2C_O}} + j\sqrt{\frac{qV_L}{2C_2}}, \tag{26}$$

where $V_L = I_1/g_{m1}$. Since $C_O > C_2$, and the linear range defined by the high-gain term is not significantly larger than V_L, the second of the two terms (the j term) typically sets most of the noise for the filter. If we are not at the Q_{max} case, we can have the case on either side of the maximum, defined as Case I in which $g_{m4}C_2 > C_Og_{m1}$ and Case II in which $g_{m4}C_2 < C_Og_{m1}$. For Case I, the noise power is within a factor of 2 of the Q_{max} case and can generally be approximated as roughly equal to the Q_{max} case (in the limit when g_{m4} is large). For Case II, we can approximate the noise power as

$$\hat{V}_{out} = \hat{V}_{out}(Q_{max})\sqrt{\frac{g_{m4}C_2}{g_{m1}C_O}}, \tag{27}$$

where $\hat{V}_{out}(Q_{max})$ is the noise level for g_{m1} at the Q_{max} level.

Figure 19 shows the output-referred noise measurement of a single C^4 and a cascade of two C^4s for various center frequencies. The noise spectrum is similar in nature

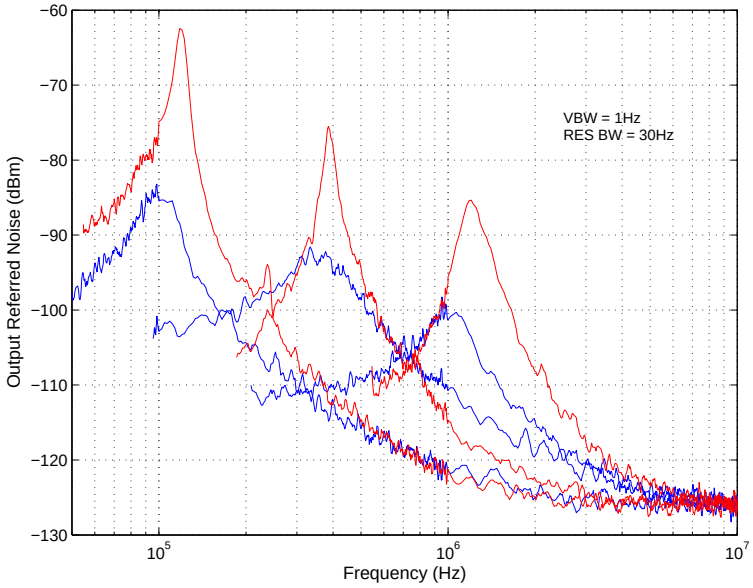


Figure 19. Output-referred noise of the C^4. This plot shows the measured output-referred noise spectrum of a C^4 and a cascade of two C^{4}s, which are both tuned to several different center frequencies.

to the frequency response of the filter, as is expected from the noise modeling experiments. Figure 19 also shows that overall noise spectrum decreases as the programmed center frequency is increased, consistant with the total noise over the bandwidth being roughly independant of center frequency. Further, we see the constant noise level expected at low frequencies, which indicates that $1/f$ noise is was not signficant over the measured bandwidth. The measured output spot-noise at 1MHz for the C^4 was found to be -100dBm (using VBW = 1Hz).

Next, we briefly consider the linearity and associated distortion terms for the C^4 filter. Using the analysis from the wideband case and focusing on subthreshold operation, the output-referred linear range from input to output is given by

$$\frac{C_T}{\kappa C_2} U_T = \frac{4Q_{max}^2 U_T}{\kappa} \tag{28}$$

and the linear range from the output to the input is V_L, which is the effective I_1/g_{m1} seen by transistor M_4 including the degeneration device (M_D). In general, the smaller of these two linear ranges sets the linear range of interest, since both are output

referred. For the differential C^4 approach, the third-order harmonic distortion at an input amplitude set at this linear range is better than -40dB for subthreshold biases; lower distortion is achieved by scaling the input amplitude appropriately, assuming the third-order power law for third-order harmonic distortion. Figure 20a shows the measurement to compute the 1dB compression point for a single C^4 and a cascade of two C^4s for two different quality factors. As expected, the linearity degrades as Q increases. The linearity for the filters (a C^4 with $Q = 2.5$ and a cascaded of two C^4s with $Q_{eff} = 5.2$) at 1MHz were -24dBm ($83mV_{pp}$) and -42dBm ($11.5mV_{pp}$), respectively. Figure 20b shows the measurement to compute the 1dB compression point for different bias values of M_D for a C^4 with low Q. It can be clearly seen that the linearity increases from -8.5dBm to -5dBm as the gate voltage of M_D is decreased from 3.3V to 1.9V. This increase in linearity comes at the cost of lowering of the low-frequency corner due to the source-degeneration effect. Thus, the current I_2 needs to be tuned to a higher value than before to achieve the same lower time constant.

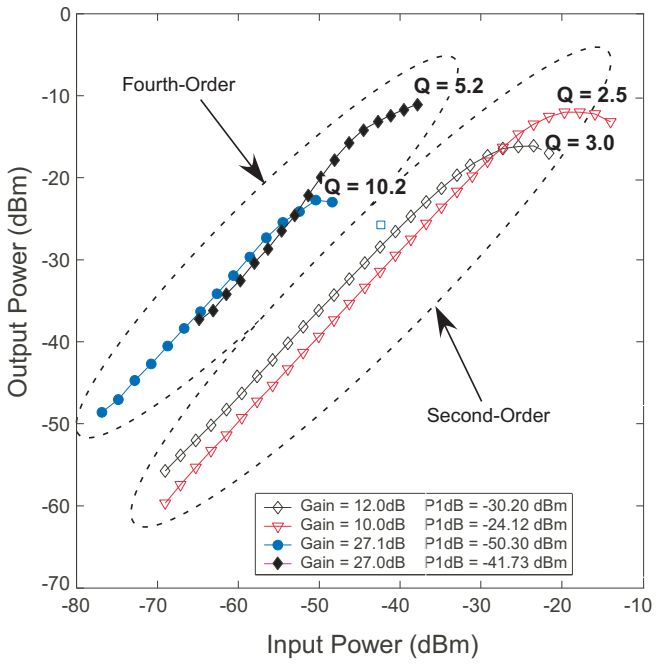
Finally, we calculate the SNR and power dissipation for this filter. Assuming the second term of (26) sets the noise for the amplifier ($= \sqrt{qV_L/C_2}$), and that V_L sets the output linear range, the SNR is

$$SNR = C_2 V_L / q. \tag{29}$$

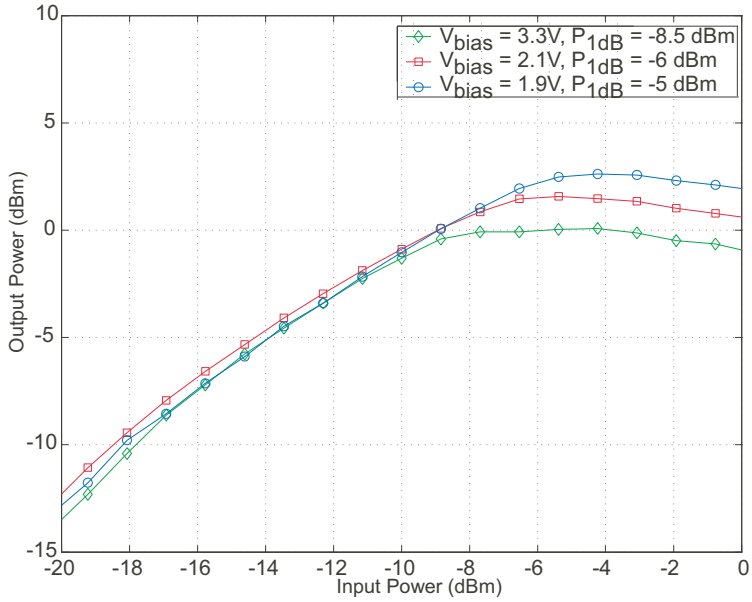
The SNR can be improved by designing for a larger Q_{max} and increasing the resulting g_{m1}, typically consuming more power as a result. The resulting power dissipation, P, for the C^4 is

$$P = 4\pi f_{center} V_{dd} Q_{max} \left(q \cdot SNR + C_O \frac{I_4}{g_{m4}} \right); \tag{30}$$

P does not include additional biasing transistors needed for a particular implementation, but their effect on power dissipation can be minimized by design. Typically, the C_O term will be less than the SNR term, because M_4 is usually biased with currents near or below threshold and because C_O is less than an order of magnitude larger



(a)



(b)

Figure 20. Linearity of the C^4. (a) 1dB compression point for different values of Q for a C^4 and a cascade of two C^{4}s. (b) Effect of the short-channel device, M_D. This increase in linearity is due to the source-degeneration effect achieved by properly biasing M_D.

Table 1. SNR and Power Dissipation for a few representative C^4 amplifier designs. V_{dd} = 3.3V, and $Q = 4$.

f_{center}	C_2	V_L	SNR	Power
20kHz	32fF	0.5V	50dB	0.11μW
20kHZ	160fF	1V	60dB	1.1μW
20MHz	64fF	0.25V	50dB	106μW
20MHZ	160fF	1V	60dB	1.06mW

than C_2; therefore, the power dissipation can be estimated as

$$P = 8\pi q f_{center} V_{dd} Q_{max} SNR \tag{31}$$

Table 1 shows the resulting SNR and power dissipation for a few representative designs of the C^4.

3.2.1 Algorithmic Design of C^4 Bandpass Filters

In this subsection, we describe how to algorithmically design a C^4 filter from a given set of specifications including linear range (V_{lin}), quality factor (Q), noise level ($\hat{V}_n$ which is directly computed from SNR), input-signal level ($V_{in,max}$), and center frequency (f_{center}). When designing a C^4 to meet given specifications, one major consideration is whether or not at $Q = Q_{max}$ the resulting ratio of C_T/C_2 sets the linear range from input to output ($U_T C_T/(\kappa C_2)$) to be larger than the specifications. If so, we start with the Q_{max} design approach; otherwise, we take an alternate approach. Further, if the resulting desired SNR is an issue, Q_{max} can be designed to be quadratically higher by the desired decrease in the noise factor.

To design the C^4 amplifier at the $Q = Q_{max}$ case, the following design equations should be used.

$$C_2 = \frac{qV_L}{\hat{V}_n^2}$$

$$C_T = 4C_2 Q_{max}^2$$

$$C_1 = 2C_2 \frac{V_{in,max}}{V_L}$$

$$g_{m1} = 4\pi f_{center} C_2 Q_{max}$$
$$\frac{g_{m4}}{g_{m1}} = \frac{C_o}{C_2} \tag{32}$$

V_L sets the output linear range. As a result, we have one free parameter available at the last step that allows us to optimize the power dissipation. Even if $C_O = C_2$, which is the minimum value for C_O, there is a minimum required amount of power, and therefore C_O weakly effects the filter operation.

For the alternate design procedure (operating in the Case I noise analysis region), the following design equations can be used.

$$C_2 = \frac{qV_L}{2\hat{V}_n^2} \text{ where } V_L = \frac{I_1}{g_{m1}},$$
$$C_T = C_2 \frac{V_{lin}}{(I_4/g_{m4})}$$
$$C_1 = 2C_T \frac{(I_4/g_{m4})}{V_{in,max}}$$
$$g_{m1} = 2\pi f_{center} C_2 Q$$
$$g_{m4} = 2\pi C_O f_{center} \frac{C_T}{QC_2} \tag{33}$$
$$P = 2\pi f_{center} C_2 V_{dd} \left[1 + \frac{C_T C_O}{C_2^2 Q^2} \left(\frac{I_4}{g_{m4}}\right)\right]$$

Again, we have a similar tradeoff for C_O, where C_O can be chosen to have a wide range for desired circuit performance, even when optimized for power.

These design approaches can be easily implemented by computer programs such as MATLAB, Excel, etc. Also, the design approach can be modified by using OTAs instead of the simple two-transistor high-gain or follower amplifiers, and achieve similar results.

3.2.2 High-Order Filter Implementation

We used the C^4 as a basic filter element in cascade to implement high-order filters. Figure 15 shows the block diagram of a tenth-order filter using these core C^4 filters.

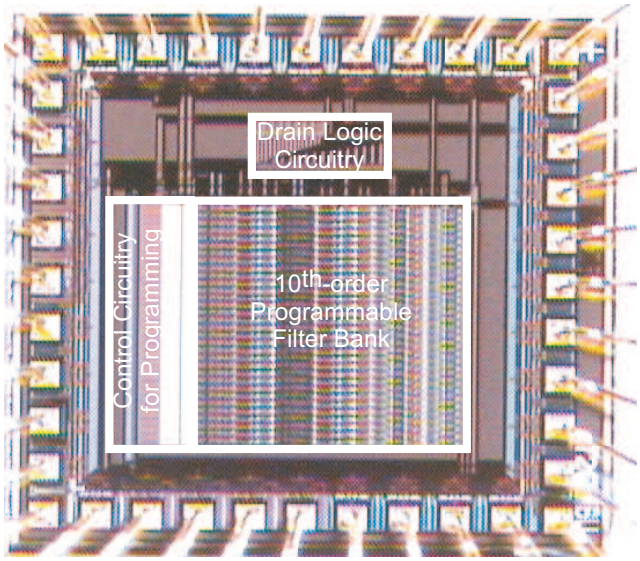


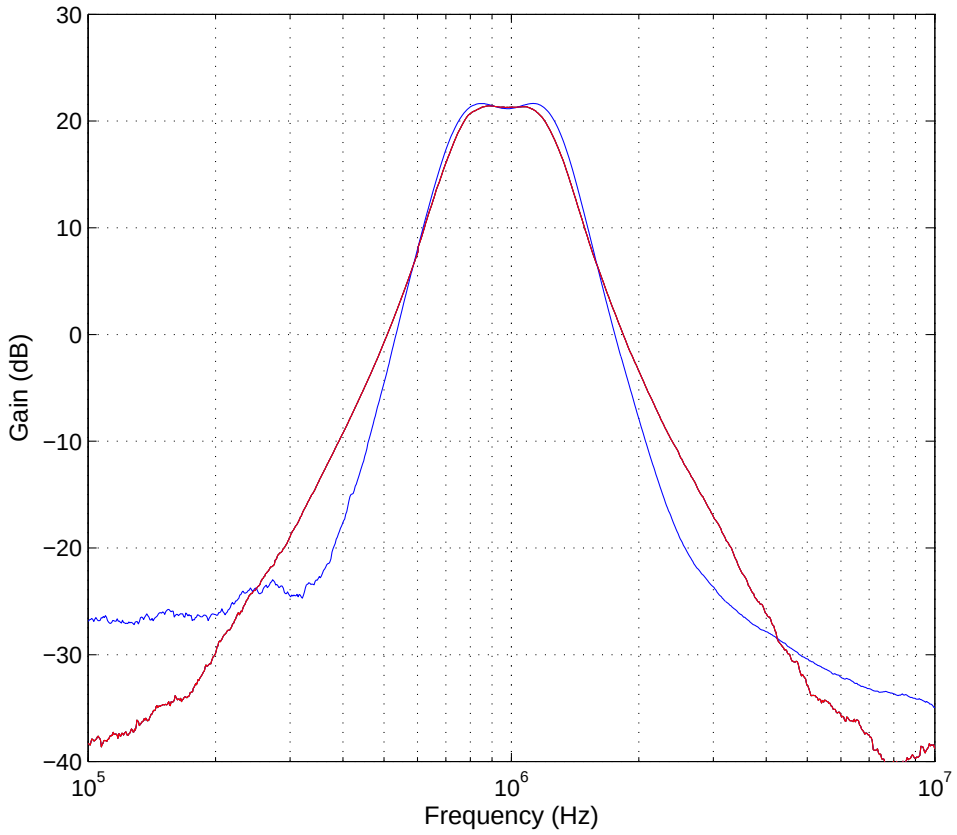
Figure 21. Die photograph of an array of 16 tenth-order filters comprised of C^4s. This integrated circuit consumes an area of only 1.5mm $\times$ 1.5mm.

These high-order filters can also be tuned to desired transfer functions, such as Butterworth or Chebyshev, after the circuit has been fabricated. The coefficients can be set by accurately programming the floating-gate currents, as will be discussed in Chapter 5.

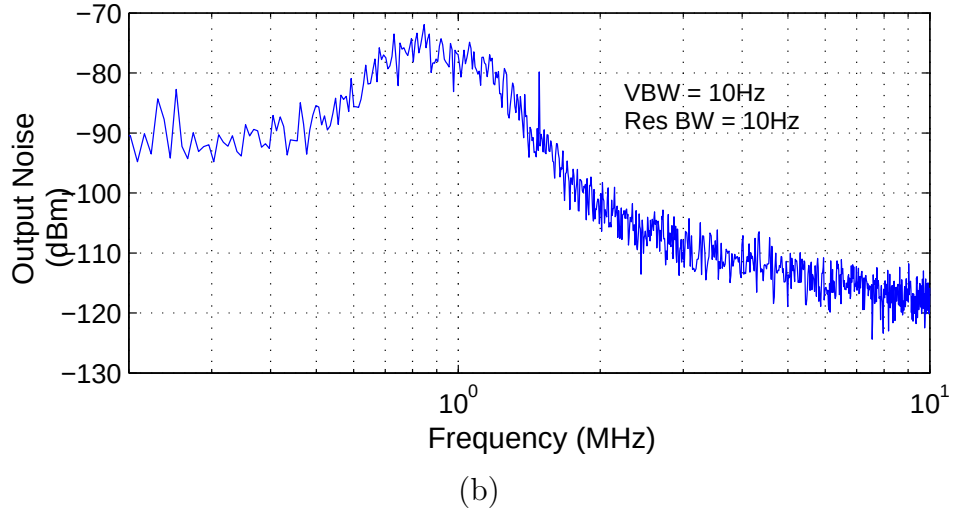
Figure 21 shows the die photograph of a chip with 16 filters that allows for the configuration of high-order filters based on the C^4. This chip can be configured as a bank of sixth-order or tenth-order filters depending on the application, and Figure 22 shows the frequency response of a sixth- and a tenth-order filter tuned to have a center frequency of 1MHz. The designed tenth-order filter was compact and power efficient. This filter can be used in a variety of filter-bank applications [9, 16].

3.3 Summary of the C^4

The C^4 has been shown to be a circuit that is useful in constructing cochlear filters as well as filters for other types of signal-processing applications. Table 2 lists the summary of performance of this continuous-time bandpass filter, as well as a cascade of two and five C^4s. The low power consumption and small real estate make the C^4



(a)



(b)

Figure 22. High-order filters constructed from C[4]s. (a) Frequency response of sixth- and tenth-order filters. (b) Output-referred noise spectrum for the tenth-order filter.

Table 2. Summary of the performance of the C^4 and a cascade of C^4s.

Parameter	2^{nd}**-order**	4^{th}**-order**	10^{th}**-order**
Frequency Range	10Hz-10MHz	10Hz-10MHz	10Hz-10MHz
Q Range	< 9	< 72	N/A
Output Noise (dBm @ 1MHz) (VBW = 1Hz)	-100dBm	-84dBm	-78dBm (VBW = 10Hz)
Total Power (with buffers)	0.1nW-15μW	0.25nW-15μW	20μW @ 1MHz
SNR @ 1MHz	86dB	72dB	55dB
Area	2.1e3μm^2	4.8e3μm^2	13.2e3μm^2

extremely attractive for building filter-bank applications.

CHAPTER 4

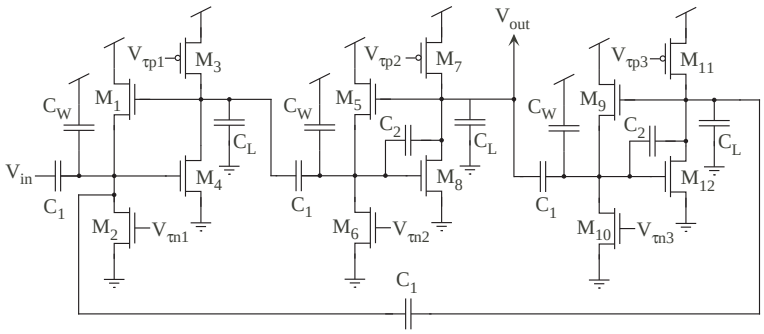
AN INITIAL SILICON COCHLEA MODEL

An initial model of the human cochlea was fabricated early in the process of this audio front-end project. This model, which we presented in [23], served as an early proof of concept to show the validity of this method of cochlea modeling. In this work, we also developed a novel circuit based on the C^4. We also used this early model in various applications including a low-power, analog noise-suppression system [8], cepstrum encoding in speech recognition [9], and a front end for phoneme recognition [10].

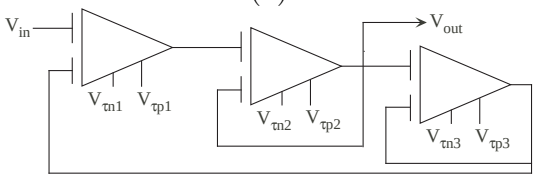
4.1 The C^4 SOS

While the C^4 has a second-order transfer function, its slopes outside the passband are first order ($\pm$20dB/decade). In keeping with the naming convention of the original cochlea models [20, 11, 14, 13], only the high-frequency responses are used to determine the "order." As a result, the C^4 will henceforward be referred to as a "first-order section" for cochlear modeling applications, even though it is truly a second-order filter.

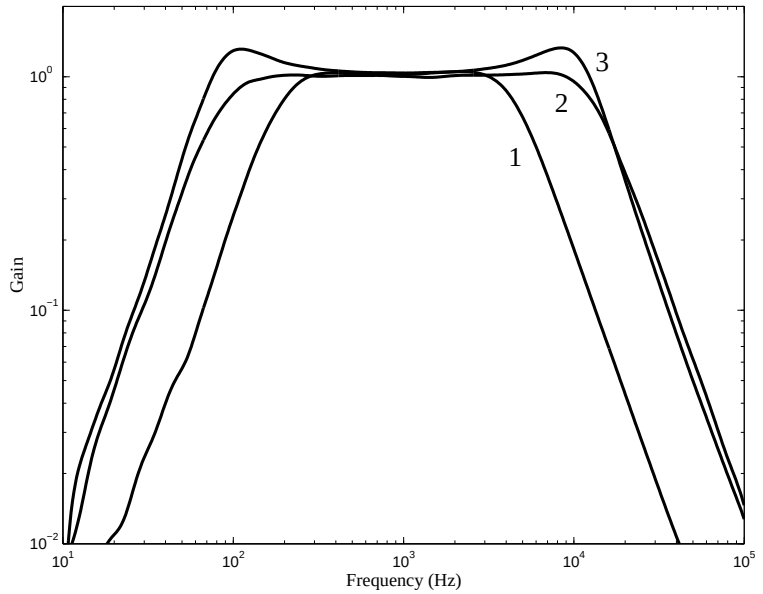
Since the C^4 is a basic first-order bandpass section, it can be used to create higher-order bandpass filters. Specifically, "second-order" bandpass sections can be designed from a combination of C^4s such that the overall response has second-order responses ($\pm$40dB/decade slopes). One such second-order section (SOS) is the C^4 SOS, which was developed before the C^4 was fully characterized and understood; as a result, this circuit did not entirely meet the desired specifications for cochlear modeling, even though it presented an interesting and useful new circuit. The following is a brief description of the C^4 SOS. Modifications to the C^4 SOS that enable it to meet the desired specifications will be presented in Chapter 6.



(a)



(b)



(c)

Figure 23. (a) Circuit schematic of the C^4 Second-Order Section (C^4 SOS). This circuit is a bandpass filter with ±40dB/decade roll offs, and it is composed of three C^4. The corner frequencies are electronically tunable and are independent of each other. The combination of the three biasing nFET's (M_2, M_6, and M_{10}) yields the low cut-off frequency, and the combination of the three biasing pFET's (M_3, M_7, and M_{11}) gives the upper cut-off frequency. (b) Shorthand notation of the C^4 SOS. The "extra" lines on the amplifier symbols emphasize the capacitive coupling for each of the amplifier stages. (c) Frequency response of the C^4 SOS. The slopes outside the passband are ±40dB/decade. For each of the three traces, the third amplifier was set to $v_{\tau n3} = 0.2$V and $v_{\tau p3} = 2.0$V. The other biases for curves 1-3 are: (1) $v_{\tau n1} = 0.41$V, $v_{\tau n2} = 0.42$V, $v_{\tau p1} = 2.45$V, $v_{\tau p2} = 2.41$V (2) $v_{\tau n1} = 0.37$V, $v_{\tau n2} = 0.38V$, $v_{\tau p1} = 2.41$V, $v_{\tau p2} = 2.37$V (3) $v_{\tau n1} = 0.33$V, $v_{\tau n2} = 0.42$V, $v_{\tau p1} = 2.37$V, $v_{\tau p2} = 2.41$V

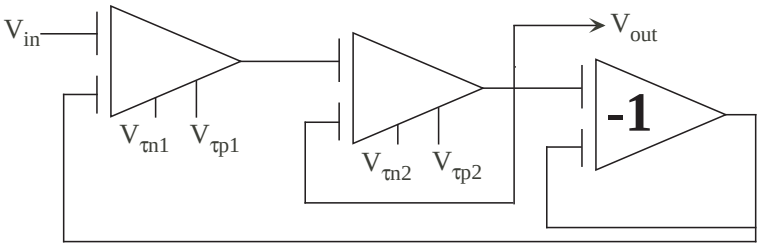
Figure 23 shows the schematic of the C^4 SOS which is a continuous-time bandpass filter with ±40dB/decade roll offs. This filter uses the C^4 as a building block and is based on the Diff2 SOS [1] and the autozeroing second-order section (AutoSOS) [41]. It is in a configuration similar to a Tow-Thomas filter, but it is composed of capacitive-based bandpass elements. The corner frequencies are electronically tunable and can be moved independently of one another.

The C^4 SOS is composed of three C^4's in the fashion of an AutoSOS [41]. The feedback capacitor of the first-stage filter is removed in order to make that stage a high gain amplifier. For each of the other two stages, the capacitors were set to $C_1 = C_2$ in order to give unity gain.

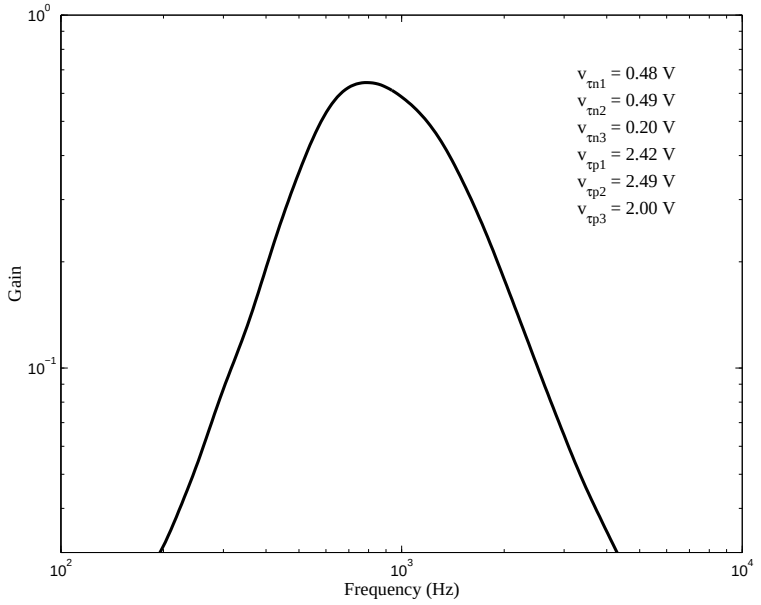
The C^4 SOS is tuned to give slopes of ±40 dB/decade by setting the third amplifier to run "fast." In essence, the third amplifier is biased so that its corner frequencies are far enough outside the frequency range under consideration that this particular stage appears to simply yield a gain of -1 (since $C_1/C_2 = 1$) over the entire range of audio frequencies. The gain of -1 thus supplies the filter with negative feedback. Therefore, the current through M_{11} is biased to a very large subthreshold current, and the current through M_{10} is biased to a very small subthreshold current.

With the time constants of the third amplifier far outside the normal range of operation, a combination of the low-frequency and high-frequency time constants of the first two filters sets the overall low-frequency and high-frequency time constants of the C^4 SOS, respectively. By adjusting the relation of $v_{\tau n1}$ to $v_{\tau n2}$ or of $v_{\tau p1}$ to $v_{\tau p2}$, the response at either corner can be tuned to have a sharp transition or even a Q peak. The response at either corner is independent of the other, as long as the two corners are sufficiently far apart. Figure 23c shows representative curves of the C^4 SOS when the corners are separated. These responses have ±40dB/decade roll offs.

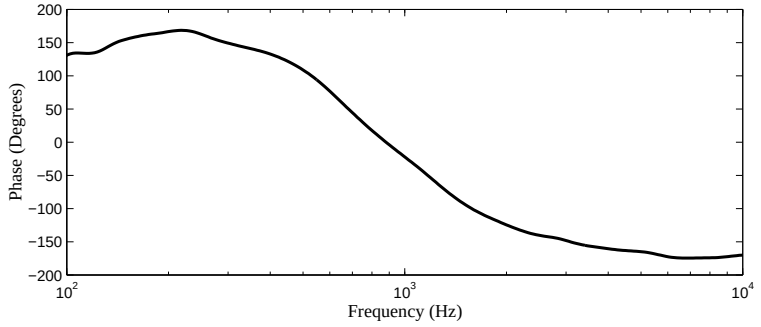
The closer the two corners of the C^4 SOS are brought together, the narrower the bandwidth becomes until a very tight band with a Q peak develops. Figure 24 shows



(a)



(b)



(c)

Figure 24. (a) Simplification of the C^4 SOS when the third amplifier is set to run "fast." This configuration provides second-order responses. (b) Magnitude frequency response for a tight bandpass filter and a small Q peak. (c) Phase response for the same bias conditions. The phase makes a sharp transition from $180°$ to $-180°$ in the region of the center frequency.

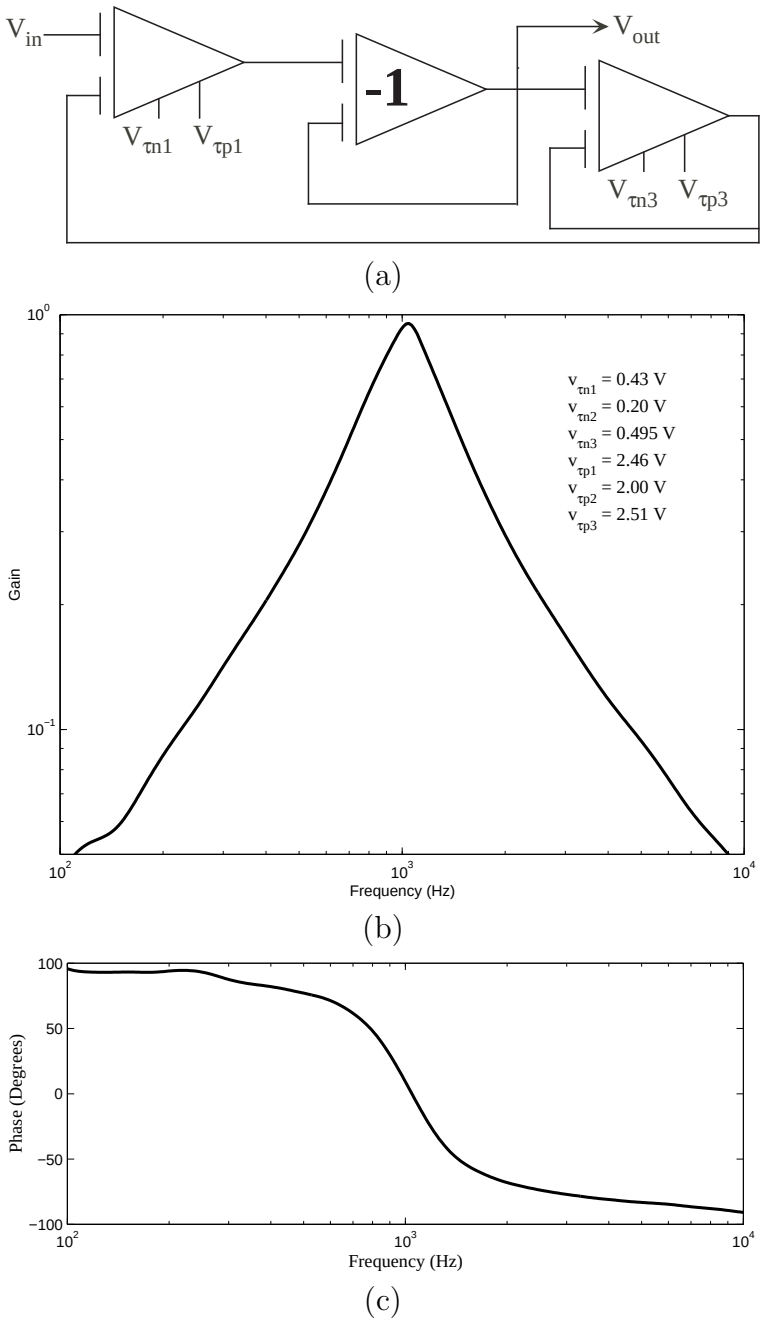


Figure 25. (a) Simplification of the C^4 SOS when the middle amplifier is set to run "fast." This configuration provides first-order responses. (b) Magnitude frequency response for a tight bandpass filter and a Q peak. (c) Phase response for the same bias conditions. The phase makes a sharp transition from 180° to -180° in the region of the center frequency.

the frequency response for a case with a small Q peak. Also included is the phase response indicating the sharp transition.

In addition to the typical mode of running this circuit, in which the third amplifier is set to run "fast," there is another useful mode of operation for the C^4 SOS. By bringing in the corner frequencies of the third amplifier to the normal operating region and by spreading out the corner frequencies of the middle amplifier so that the second amplifier is set to run "fast," the frequency response of the C^4 SOS is of the form shown in Figure 25. Running in this mode of operation, the C^4 SOS has larger Q peaks for narrow-bandwidth responses, but the cost is that the roll offs are only ±20dB/decade. Even when thinking about cochlear responses, this type of response is still valid because it models the response of the basilar membrane in terms of displacement, instead of velocity.

4.2 Resistor-Based Cochlea Model

We built an array of 32 C^4 SOSs, as is illustrated in Figure 26b, which we first presented in [23] and which represents the first silicon cochlear model using a bandpass-filtering approach. Six tapped resistive lines (one for each bias transistor) were used to space the center frequencies exponentially. The linear voltage division of the resistive lines translates into exponential changes in transistor currents when biased in the subthreshold regime. The filters were biased with both ±20 and ±40dB/decade slopes. First-order slopes show the response of the basilar membrane in terms of displacement, while the second-order case represents the basilar memebrane's velocity. The advantage of the second-order responses is that potentially less coupling will be required to mimic the human cochlea.

Figure 27a shows the response of the array when each stage was biased to have ±40dB/decade slopes. Also included is Figure 27b which shows the response of the array for the case when each stage was biased to have ±20dB/decade slopes.

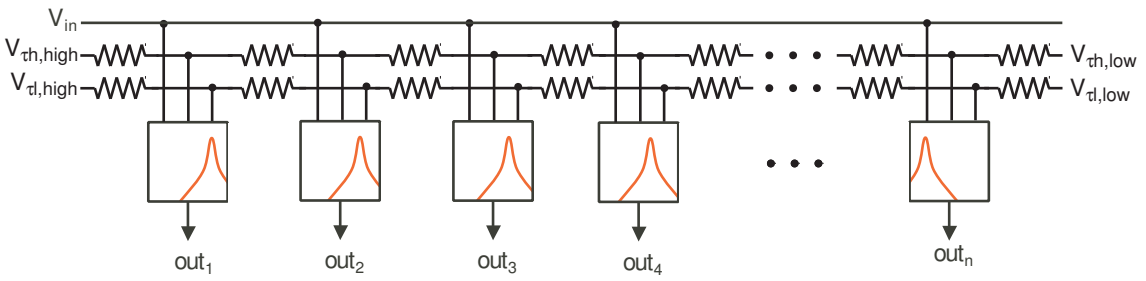
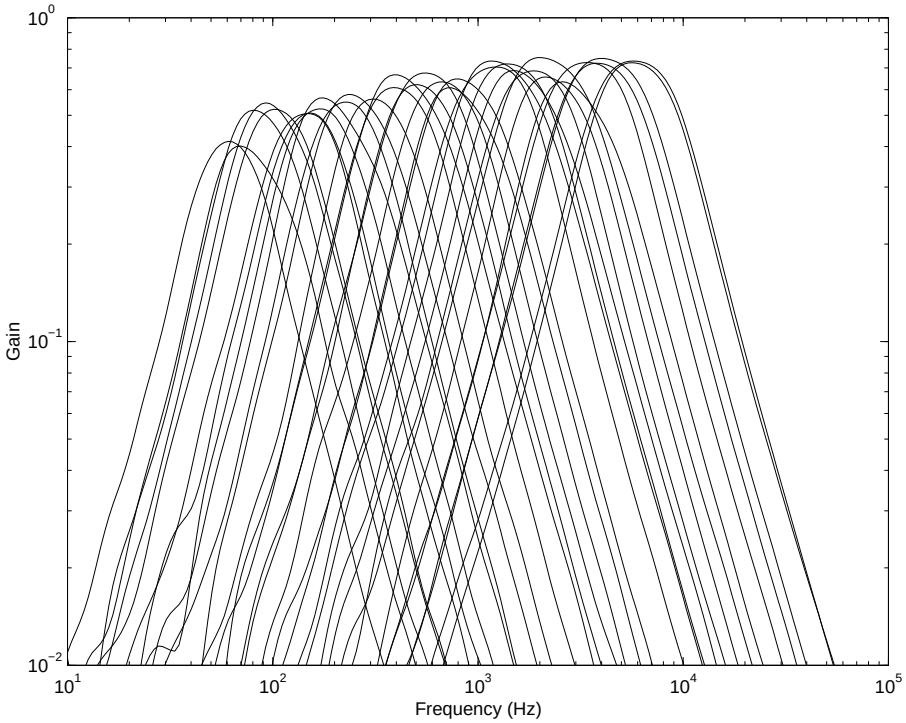


Figure 26. Schematic of the array of bandpass filters used for cochlear modeling. This array is tuned from high frequencies to low frequencies, as is the human cochlea. Large resistive lines are used for exponentially biasing each C^4 in the array. Two resistor lines are required for each C^4; thus, a C^4 SOS requires six resistor lines.

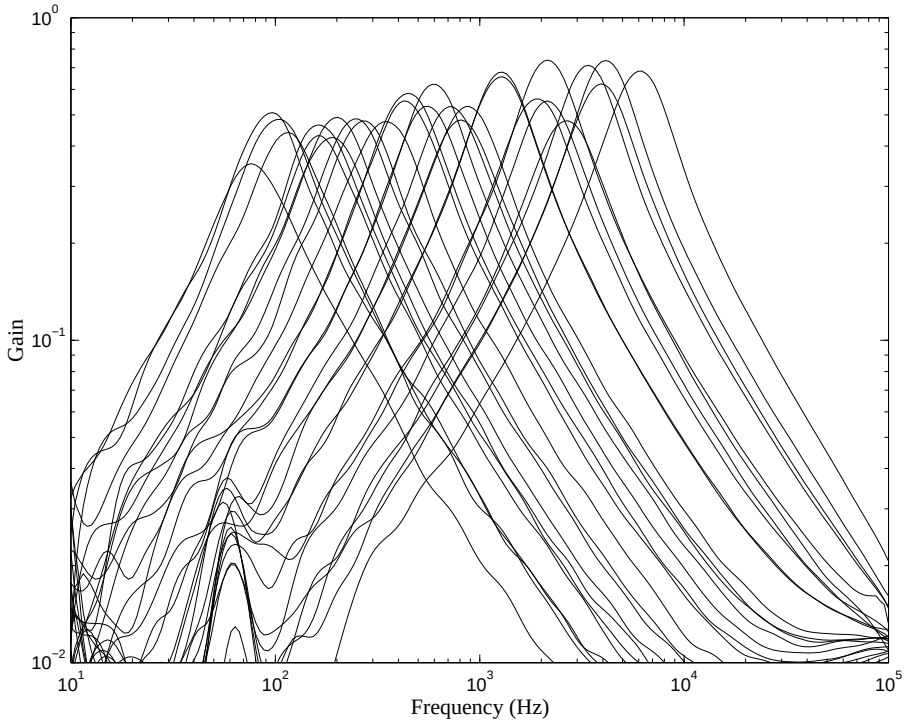
Both plots of Figure 27 illustrate the significant effects of device mismatch resulting from imperfections in the fabrication process. Because of variances in the resistor, capacitor, and transistor sizes from the designed values, the time constants were not as perfectly tuned as would have been desired. As a result, the time constants were unevenly spaced, thus resulting in uneven center-frequency spacing and variances in the gain of the individual filter taps.

However, even though this system was limited by processing imperfections, the plot of Figure 28 shows that the center frequencies of each stage were still spaced monotonically and relatively evenly. Even without using any special matching techniques, this cochlea model had correctly spaced stages. Creating nearly linear traces like this was a difficult task for those who tried to use the cascade of lowpass filters as a cochlea model [11, 12]. There is definitely merit to this resonance-based model.

While there is much room for improvement with this cochlea model, especially in the areas of having filters with more cochlea-like responses (having higher Q peaks) and having evenly spaced filters, this filter-bank system has been shown to be very useful as a front end to audio systems. For example, the cochlea model was used as a front end for a noise-supression system [8], for cepstrum encoding in speech recognition [9], and for phoneme recognition [10]. In general, this model seems to work well in virtually any system needing an analog signal processing block [42].



(a)



(b)

Figure 27. a) Frequency response of the array with second-order slopes. (b) Frequency response of the array with first-order slopes.

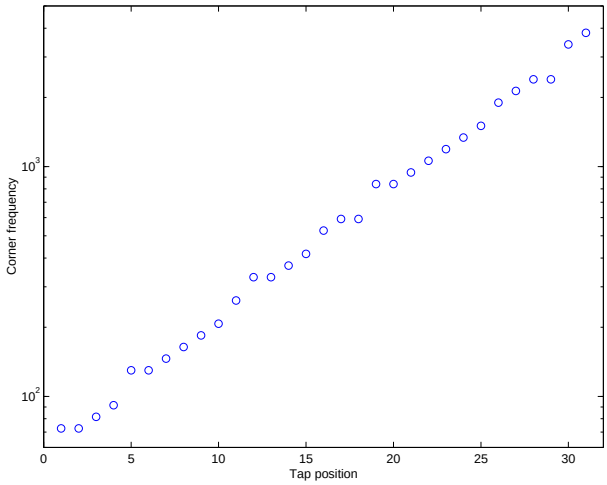


Figure 28. A plot of the center frequencies for each of the taps in the filter bank. The center frequencies are spaced monotonically. These data are from the case in which the stages were biased to have ±40dB/decade slopes. Also, this spacing shows the filter taps going from low to high frequencies, which is opposite from what happens in the real cochlea. This ordering was done to illustrate that ordering is not important in this bandpass-array approach.

While this filter bank is a useful system block, it is clear from the data in Figure 27, that this filter bank was not perfectly tuned. However, the filter bank would serve as a better decomposition block if it were so. The major source of errors in the tuning of the filter bank is the mismatch of the circuit devices, namely the resistors, capacitors, and transistors. Since the bias currents were not likely perfectly exponentially spaced and since the error in the capacitor sizes produced time constants different from the desired values, the two corner frequencies of the C^4 SOS tended to be spread around the respective center frequency differently for each C^4 SOS, thus leading to non-uniform center-frequency spacing and differences in the midband gain.

No matching techniques, such as common-centroid layout, were used in the design of this filter bank, and very small devices (maximum capacitor of 37fF and maximum transistor length and width of 4.8μm) were used. Therefore, this degree of mismatch was not unexpected. However, this bandpass filter-bank approach performed comparably to early cochlea models in which matching techniques and much larger device sizes were employed [12, 33].

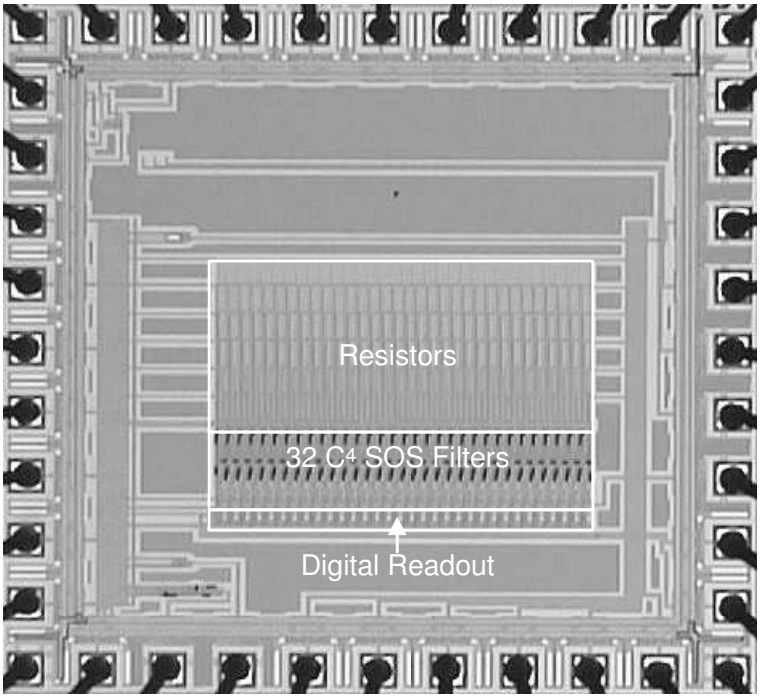


Figure 29. Die photograph of the resistively biased filter bank. Total area dimensions are 1.5mm $\times$ 1.5mm. The resistor lines consume a large amount of real estate yet still do not achieve the desired results in terms of center-frequency accuracy.

The resistors, which are shown in the die photograph of Figure 29, consumed a large amount of real estate yet still did not grant the performance that was desired. Using floating-gate transistors to bias the filters indtead of large resistor lines will reduce the overall real estate while simultaneously improving the performance by negating the effects of mismatch. Additionally, floating-gate transistors provide the added flexibility of programmability. The following chapter gives a general overview of floating-gate transistors and the method of accurately programming them.

CHAPTER 5

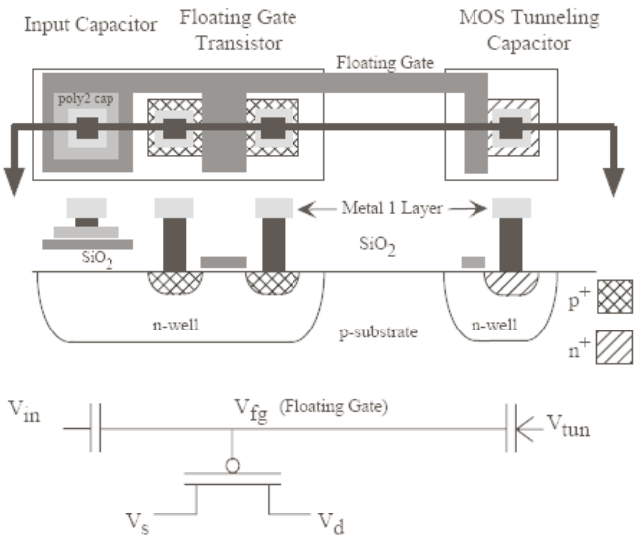
FLOATING-GATE TRANSISTORS FOR ANALOG PROGRAMMABILITY

One of the primary problems with the initial bandpass cochlea model, as well as the previous cochlea models, was that the corner frequencies of the filters could not be accurately controlled. This lack of control is due to mismatch of the devices, especially the transistors used to bias the C^4s and their resulting currents. To achieve accurate control over the biasing currents, floating-gate transistors are utilized since they allow precise control over their currents due to programmability.

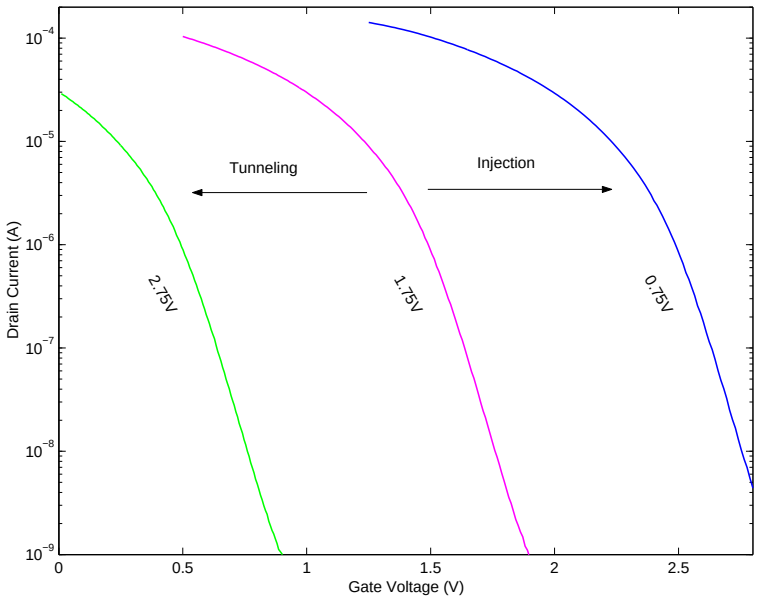
5.1 Floating-Gate Transistor Overview

A single floating-gate (FG) transistor, shown in Figure 30a, is simply a standard MOSFET device with only capacitors connected to the gate. Since the gate is electrically isolated due to oxide completely surrounding it, the charge on the gate is fixed and is responsible for establishing the amount of current flowing through the transistor. While the charge on the gate will not change on its own, that amount of charge can be modified by processes such as UV photo injection, Fowler-Nordheim tunneling, and hot-electron injection. The last two are the primary means of programming floating-gate transistors to precise currents [43].

Through the process of electron tunneling, a large voltage is placed across a MOS capacitor. As this large tunneling voltage is increased, the effective width of the barrier is decreased, thus allowing electrons to breach the gap without adversely affecting the insulator. Tunneling is used to remove electrons from the floating gate in a controlled manner and, thus, raises the effective threshold voltage (referenced to V_{dd}), as is shown in Figure 30b.



(a)



(b)

Figure 30. (a) Floating-gate pFET. The gate is electrically isolated from the rest of the circuit by connecting only capacitors to it. The charge on the floating gate determines the current that flows through the transistor. This charge can be modified by programming using tunneling to remove electrons or hot-electron injection to add electrons to the floating gate. (b) Movement of the threshold voltage. Tunneling increases the threshold voltage (with respect to V_{dd} since this is a pFET device). Injection decreases the threshold voltages.

Whereas tunneling is used to remove electrons from the FG, hot-electron injection is used to add electrons in a controlled manner. Hot-electron injection has two requirements. First, an appreciable amount of current must be flowing through the device. Second, a large source-to-drain voltage must be placed across the transistor. When both of these criteria are met, holes in a pFET flowing through the channel can build up sufficiently large energy to impact ionize an electron-hole pair. The resulting electron can have enough energy to pass through the insulator and onto the FG, thus adding electrons to the gate and therefore lowering the effective threshold voltage (Figure 30b). Since process-control parameters are set to stop injection from occurring in n-channel devices, only p-channel devices are used for FG programming [44].

5.1.1 Programming Precision

Figure 31 shows the programming accuracy that is presently achievable [45, 46]. The accuracy to which a FG transistor can be programmed to meet a target current depends on the smallest drain current change that can be programmed onto a FG device. Assuming that the FG transistor is operating in the sub-threshold regime, the programming precision can be determined as follows. The drain current is given by

$$I = I_o e^{-\kappa V_{fg}/U_T} e^{V_s/U_T}. \tag{34}$$

where κ is the capacitive ratio coupling from the gate to the surface potential and U_T is the thermal voltage. For a change in the gate voltage, ΔV_{fg}, a change in drain current, ΔI, results, and the net programmed drain current of the device is given by

$$I + \Delta I = I_o e^{-\kappa (V_{fg} + \Delta V_{fg})/U_T} e^{V_s/U_T}. \tag{35}$$

Dividing (35) by (34) gives

$$\frac{\Delta I}{I} = e^{-\kappa \Delta V_{fg}/U_T} - 1. \tag{36}$$

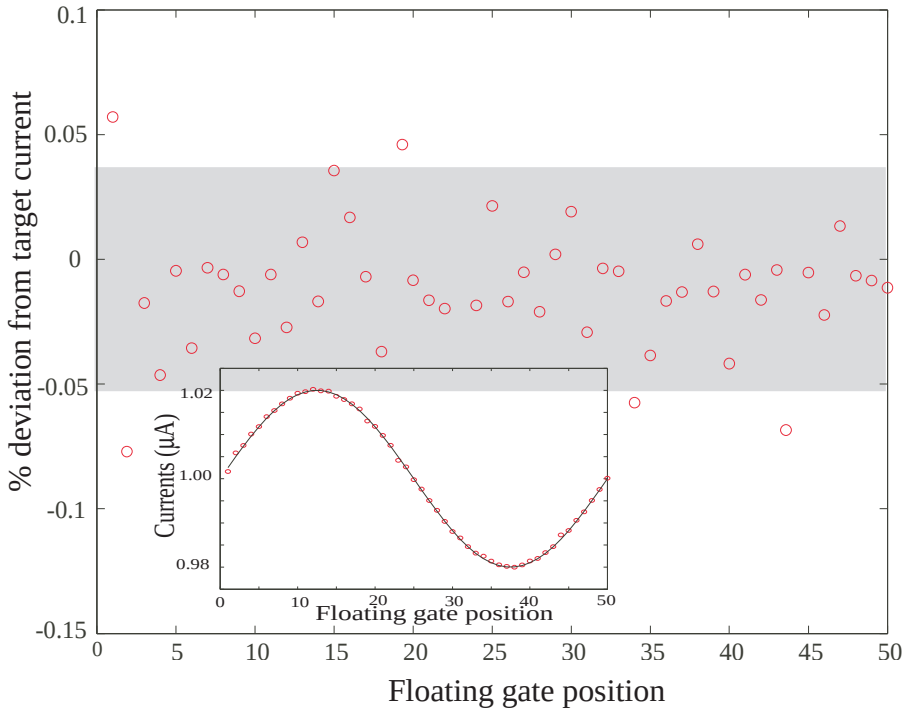


Figure 31. Floating-gate programming precision. Programming a 20nA sinusoid riding on a DC value of 1μA is shown along with the percentage error between the programmed current and the desired target. As can be observed, an error of $\pm 0.05\%$ has been achieved.

The change in FG voltage is related to the programmed FG charge by

$$\Delta V_{fg} = \frac{\Delta Q}{C_T} \tag{37}$$

where C_T is the total capacitance connected to the floating-gate node and ΔQ is the programmed charge. Using (37) in (36) yields the achievable change in drain current from programming relative to the initial drain current, and this ratio is given by

$$\frac{\Delta I}{I} = e^{-\kappa \Delta Q / U_T C_T} - 1. \tag{38}$$

As (38) indicates, the achievable precision is directly proportional to the charge that can be reliably transferred onto the FG and inversely proportional to the total FG capacitance. As an example, using the theoretical minimum for charge transfer, which is equal to that of a single electron, a FG capacitance of $C_T = 16$fF (a small

device), $\kappa = 0.7$, and $U_T = 25\text{mV}$, then a single electron change results in an accuracy of 2.8×10^{-4} (12 bits) over the entire sub-threshold range of 6-8 decades. If, however, the capacitance is increased by a factor of 10, the accuracy improves to 2.8×10^{-5}, or 15 bits.

5.1.2 Floating-Gate Charge Retention

Charge loss in FG transistors falls under two categories that occur due to different physical processes including short-term drift that is observed immediately after programming and long-term charge loss that occurs over years. The short-term drift in FG charge has been attributed to the interface trap sites settling to a new equilibrium. Also, it has been observed that the drift is proportional to the amount of charge that is programmed onto the floating gate. For instance, using the threshold voltage of the device as an indicator of the programmed charge, the short-term drift in the threshold voltage is proportional to the difference between the programmed threshold voltage and its initial value [47].

Long-term charge loss in FG transistors occur due to a phenomenon known as thermionic emission [48, 49, 50]. The amount of charge lost is a function of both temperature and time. Data extrapolated from accelerated temperature tests on floating-gate transistors, in which FG transistors have been exposed to high temperatures ($> 125°\text{C}$) for prolonged periods of time, indicate FG charge loss of $< 1\%$ over a period of 10 years [47] thereby demonstrating excellent charge retention.

5.2 Direct Programming of Floating-Gate Transistors

To program a large amount of floating-gate devices, as is required for a programmable filterbank, FG transistors are arranged in an array for ease of programming, as shown in Figure 32 [43]. In this configuration, all the drains of the FG transistors within a row are connected together and all control-gate voltages within a column are also

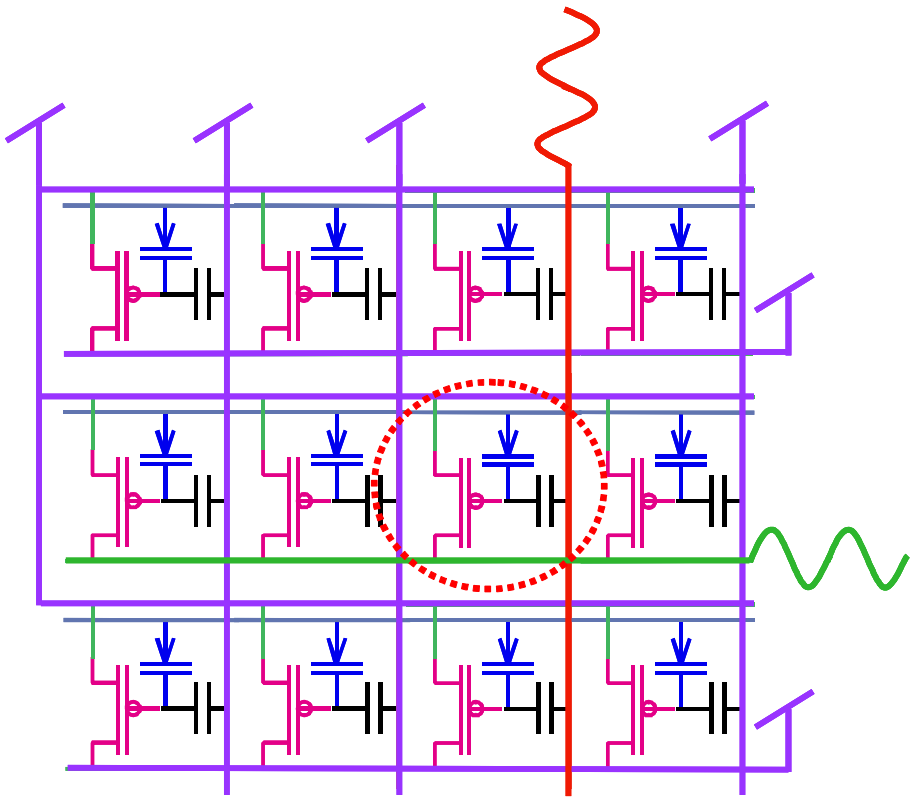


Figure 32. Array of floating-gates transistors that are precisely programmed by injection. Floating gates for large systems are typically arranged in arrays for easy programming. Injection is used to get selective and precise currents. After an element is selected to be programmed, all the columns beside the one with the selected element are connected to V_{DD} and all the rows beside the one with the selected element are also conected to V_{DD} to turn off the current in all the other transistors. Then a gate voltage is applied to the selected element, and the drain is pulsed to a low voltage so that injection occurs only on the selected element.

connected together. While tunneling can be used to program currents accurately, selectivity is not completely controllable in this arrangement. As a result, the tunneling operation is reserved for globally "erasing" the charges stored on the floating gates.

However, hot-electron injection allows complete selectivity of an individual element and is used for precise and accurate programming of FG arrays [43]. Selecting a particular device for injection involves connecting all unselected rows of drain lines to V_{dd} and all unselected columns of gate lines also to V_{dd}. As a result, only a single device will have an appreciable current flowing through its channel at any given time. Therefore, only this single device will meet both criteria for injection to occur, which

are that a current must flow through the device and that the device must have a large source-to-drain potential. The gate and drain of this selected device are both pulsed down so that both injection criteria are met, and electrons are added to the FG until the channel current matches the desired value.

Each transistor can be selected in this same manner and programmed to the desired current. When all the currents have been set to the desired values, the terminals of the transistors are connected to the rest of the circuit in which they are operating, and they behave as fully functional transistors. Figure 31 shows that using this array programming procedure, a high-degree of programming accuracy can be achieved.

When using this direct method of programming, thus named because the desired current is directly programmed into the appropriate FG transistor, the FG transistor must be physically removed from the circuit in which it is being used for a programming phase. The FG transistor is therefore switched between programming circuitry and its respective circuit via a multiplexer. Often this multiplexing is done with a simple transmission-gate (T-gate) switch. While this direct method of programming allows the programmed current to be observed, and can thus yield a high degree of accuracy, the added selection circuitry adds parasitics that can hamper the circuit's overall performance.

5.3 Indirect Programming of Floating-Gate Transistors

Programming FG transistors has previously required using transmission gates (T-gates) to disconnect each FG transistor from its circuit for a programming phase and then reconnecting it for a run-time phase [43]. However, the addition of a 2-to-1 multiplexer for every FG to be programmed can be costly. The process of disconnection can decrease the maximum speed of operation and overall accuracy while

also increasing the required real estate and necessary supply overhead. To circumvent the problems associated with detaching the FG transistor, we introduce a new, non-invasive method of programming that eliminates the need for disconnection and instead uses an indirect method of programming.

The concept of indirect programming of floating-gate transistors is illustrated in Figure 33a-b. An early discussion with preliminary results was included in [51], and a more thorough treatment of indirect programming has been included in [52]. With this indirect programming technique, multiple MOSFETs share a common floating gate. One pFET is connected to the programming structure while the source and drain of the other transistor are connected to the respective circuit. The first pFET is programmed with hot-electron injection and tunnelling using the method of [53] . Since the charge on this "programmer" pFET is modified, the current of the other transistor (the "agent") will also be set.

5.3.1 Motivation for Indirect Programming

To illustrate the usefulness of this indirect programming method, Figure 34a shows the FG current mirror introduced in [54] for perfectly matching the two leg currents. The full schematic of this current mirror is actually given by Figure 34b, and the increase in complexity is clearly evident. The additional resistances and capacitances introduced by the eight T-gates, used to break the FG transistors out of the mirror for programming, seriously hamper the performance of the current mirror, especially at high frequencies. The simple two-transistor current mirror becomes a complex 18-transistor circuit.

The use of indirectly programmed FG transistors simplifies the pFET current mirror to that of Figure 34c. Now, only a minimal amount of disconnects need to be included. Only two cascoding transistors and a single T-gate are used, and the cascoding transistors serve the dual purpose of isolating the FG transistor and enhancing the current response of the mirror.

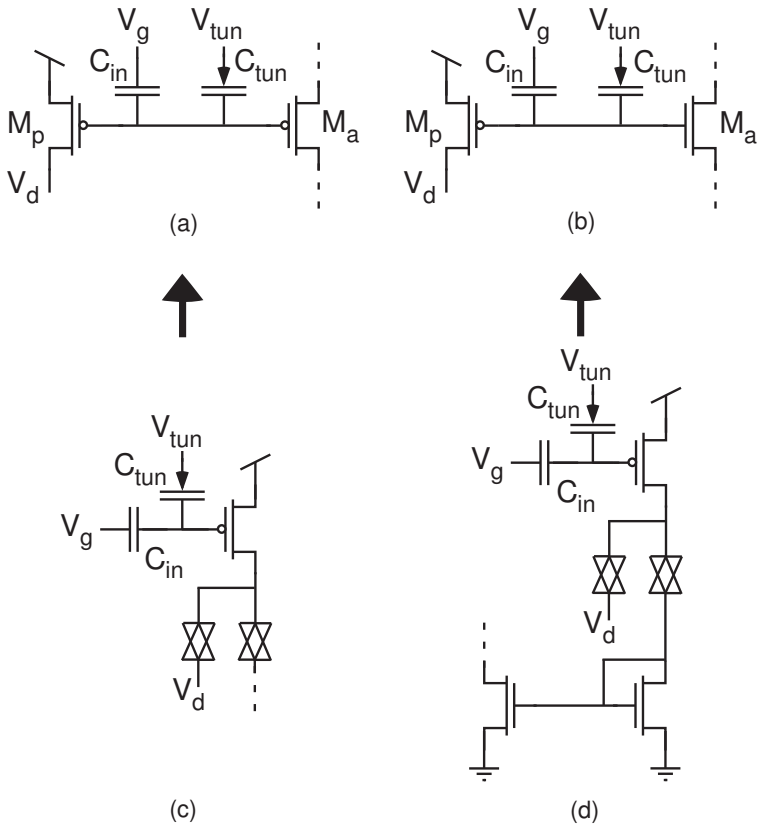


Figure 33. (a) Programming structure of a pFET indirectly programming another pFET. The programmer transistor, M_p, is connected to the external programming structure and is actively programmed via hot-electron injection. The agent transistor, M_a, is connected to its circuit (represented by the dotted lines) and is passively programmed. (b) Programming structure of a pFET indirectly programming an nFET. (c) Direct method of programming a pFET. Direct programming requires disconnecting the pFET from the rest of the circuit with transmission gates (T-gates). This schematic represents a best-case scenario in which only two T-gates are required. For some applications, two T-gates each at the source and gate would also be required. (d) Direct method of programming an nFET. Direct programming requires programming the current in a pFET and then mirroring that current into the nFET that is connected to the circuit. For all shown FG transistors, V_{tun} is used for tunneling the FG node or is set to a constant DC voltage in run mode equal to the voltage used when measuring the current.

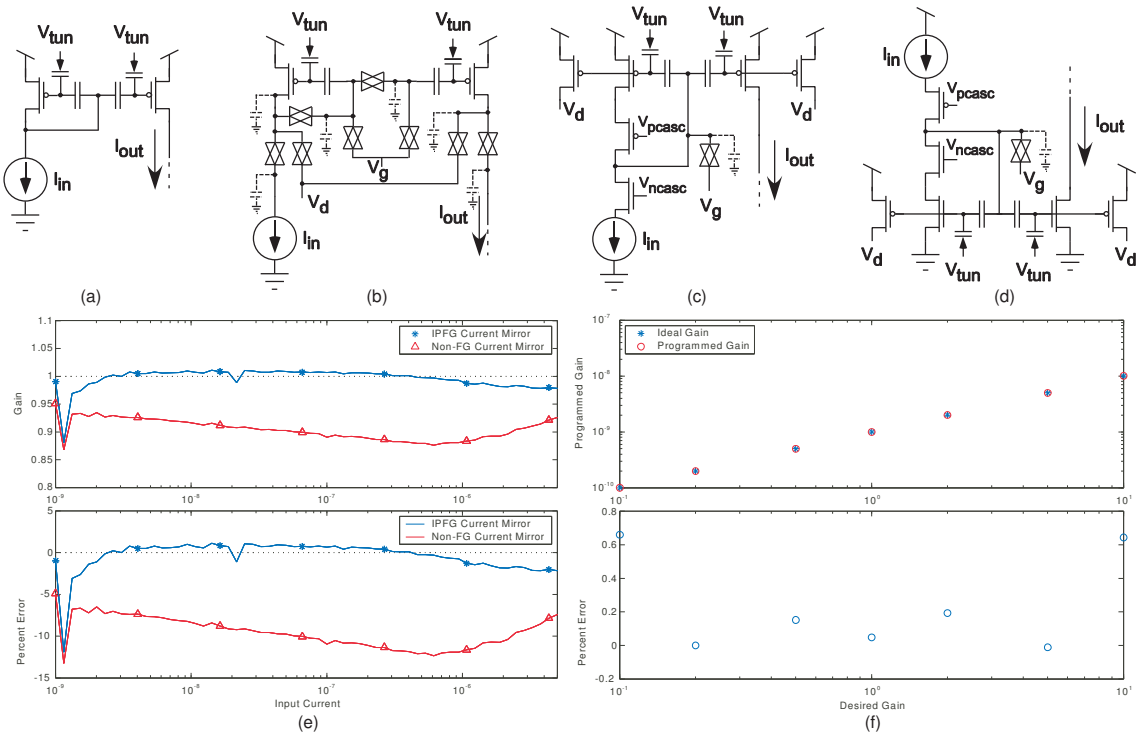


Figure 34. (a) Floating-gate transistors for offset removal in a current mirror. (b) Implementation of the FG current mirror using direct FG programming techniques. To allow complete disconnection of each FG transistor for programming, many T-gate switches must be used which add parasitic capacitances (shown in dashed lines) and resistances. These switches increase the required area and supply headroom while concurrently degrading the operational performance. (c) Implementation of the FG current mirror with the indirect-programming technique. The use of indirectly programmed transistors greatly reduces the complexity of the circuitry and minimizes the parasitics. The two cascode transistors are included for both improved performance and also for isolation of the gate voltage for programming. (d) Implementation of an nFET FG current mirror with indirect programming. This current mirror is a simple design, whereas the construction of an nFET programmable current mirror using the direct programming method is virtually impossible. (e) Indirectly programmed floating-gate (IPFG) nFET current mirror data. The charge on the two floating-gate nodes of (d) were normalized, causing the current gain to be nearly unity for a large range of current values. Data from a non-FG nFET current mirror are also included, and the improvements with the FG version is clearly evident. (f) Various current gains programmed in the IPFG nFET current mirror. Programming with IPFG transistors allow the current gain to be modified after fabrication. These current gains apply to all subthreshold currents. For above threshold current levels, the gains apply only for small signal deviations.

Precise programming of nFETs with hot-electron injection is virtually impossible due to process-control techniques that specifically work to avoid nFET injection [55]. When an nFET is to be used as a precise current source with FGs, a pFET is programmed, and that current is mirrored into the nFET current source, as shown in Figure 33d. Therefore, creating a programmable nFET current mirror with the direct method of programming is no simple task.

The process of programming an nFET is more explicit with indirect programming. Since an nFET and pFET can share the same floating gate, the nFET current is set by programming the pFET. This technique allows the construction of a programmable nFET current mirror (Figure 34d) that is completely analogous to the pFET version of Figure 34c.

Figure 34e-f shows the benefits of using not only a floating-gate programmable current mirror, but also an indirectly programmed version. The data from these plots were all obtained from an nFET version of the programmable current mirror. Data from a pFET version of an indirectly programmed current mirror has similar results, but only the nFET version, which was not previously capable of being built, is shown here for simplicity.

Figure 34e shows that the result of normalizing the charge on the two floating nodes in the current mirror allows the current mirror to perform very close to the ideal. This nFET version of an indirectly programmed floating-gate (IPFG) current mirror was constructed using identically sized FG transistors. Therefore, normalizing the charge on the two floating nodes resulted in identical currents flowing through both legs of the IPFG current mirror. Since the subthreshold current flowing through an FG transistor in saturation is given by

$$I = I_0 e^{\kappa V_{FG}/U_T} e^{-V_s/U_T} e^{\kappa V_d/V_A}, \tag{39}$$

the current gain, I_{out}/I_{in}, is

$$\frac{I_{out}}{I_{in}} = \frac{I_0 e^{\kappa V_{FG,out}/U_T} e^{-V_s/U_T} e^{V_d/V_A}}{I_0 e^{\kappa V_{FG,in}/U_T} e^{-V_s/U_T} e^{V_d/V_A}} = \frac{e^{\kappa V_{FG,out}/U_T}}{e^{\kappa V_{FG,in}/U_T}} \approx 1 \tag{40}$$

assuming that the drains are at similar potentials. In these subthreshold equations, U_T is the thermal voltage, κ is the capacitive ratio coupling from the gate to the surface potential, and V_A is the Early voltage [1].

Figure 34e shows that the gain can, indeed, be made very close to unity by programming identical charges to the two floating nodes. Also included are data from a standard two-transistor current mirror showing that the percent error from the input to the output is $7-10\%$ over a wide range of input currents. This degree of mismatch is not unexpected for small-sized transistors $\left(\frac{W}{L} = \frac{2.4\mu m}{1.2\mu m}\right)$ such as these [56].

In addition to normalizing the FG charge for a unity-gain current mirror, this IPFG current mirror allows the gain to be set after fabrication by programming different charges to the two floating nodes. Figure 34f shows measurements of the current mirror being programmed to a variety of gains. These gains were well within 1% accuracy. While (40) allows the IPFG current mirror to achieve unity gain over a wide range of current levels, this same relationship will only allow the current mirror to achieve the desired non-unity gains while both transistors stay in the subthreshold region. Once one transistor enters moderate or strong inversion, the exponential relationship of (39) no longer holds, and the gains will degrade from their programmed values. Therefore, the baseline current for measurement in Figure 34 was a subthreshold current (1nA).

This current mirror example shows several of the distinct advantages of indirect programming over previous methods. These advantages, and others that have not yet been mentioned, are summarized as follows. Indirect programming of floating-gate transistors

- Allows nFET programming

- Decreases the number of poles / parasitic capacitances for faster operational speeds
- Decreases resistance
- Decreases minimum supply headroom
- Reduces transistor count / real estate
- Permits run-time programming / calibration

5.3.2 Indirect Programming of pFET Transistors

The most basic method of indirect programming uses injection in the programming pFET to set the current in the agent pFET and tunelling for erasing that current. The programming pFET can be placed in a large FG array similar to that shown in Figure 32 and selected and programmed in the fashion of [43]. The output of the agent will be a scaled version of the programmer, assuming the drain and source potentials of the two devices are similar. Scaling is due to $\frac{W}{L}$ ratios and any mismatch between the two devices. Figure 35a shows the I-V characteristics for a gate sweep of both the programmer and the agent, which are identically sized devices ($\frac{W}{L} = 2$). Typically, the agent current is unobservable, but these data are from an isolated pFET-pFET pair sharing the same FG that will be used for characterization purposes.

Assuming that the sources and drains of the two transistors are at similar potentials is not always valid. Figure 35b shows the effects of varying the source potential of the agent. With both transistors in the subthreshold regime, varying the programmer current yields approximately a $1 : 1$ change in the agent current. The exact relationship is a ratio of the subthreshold slope, $\frac{\kappa}{U_T}$, of the two transistors, which should be very closely matched due to their same orientation and close proximity in layout.

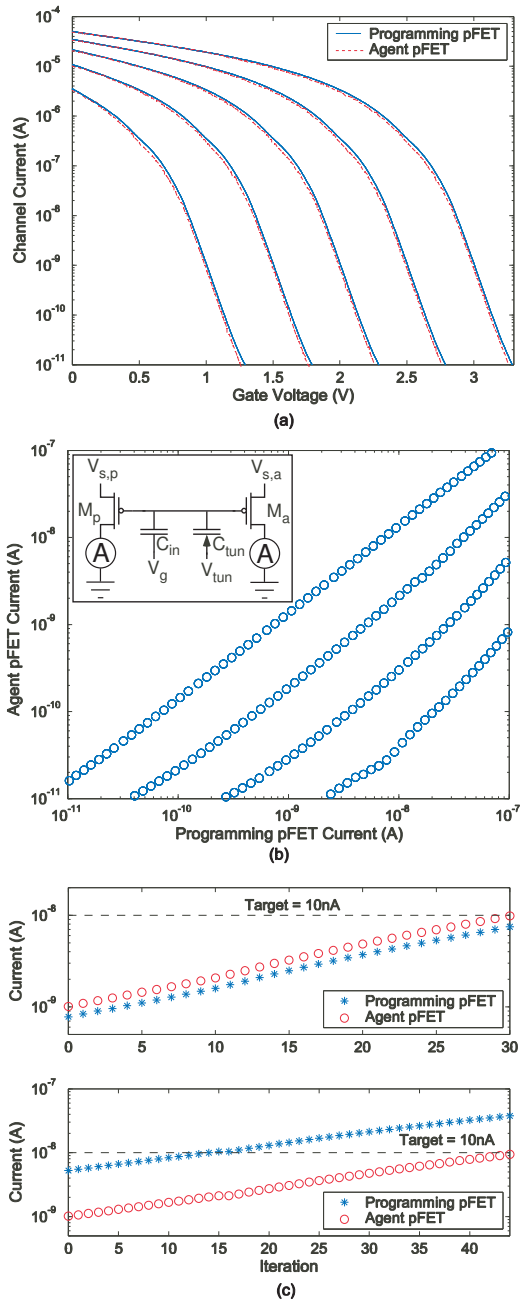


Figure 35. (a) I-V characteristics of an indirectly programmed pFET $\left(\frac{W}{L} = 2\right)$ and its programming pFET $\left(\frac{W}{L} = 2\right)$. The currents were measured simultaneously through two identical picoammeters using the schematic shown in this figure. Typically, the current through the agent is unobservable, but these data are from an isolated pFET-pFET pair used for characterization. (b) Schematic for the measurements and the ratio of the programming pFET current to the agent pFET current for various values of $V_{s,a}$. The slope of each trace begins to differ from unity at low current levels due to measurement limitations. At high current levels, the slope differs from unity since the programming pFET leaves subthreshold sooner than the agent pFET as $V_{s,a}$ is increased. (c) Programming the agent pFET to a target. (Top) Programming when the sources are at similar potentials. (Bottom) Programming when the agent pFET's source has been lowered below the programmer pFET's source potential.

When programming the agent current to a desired value, only the programmer current is observable. Therefore, measurement of the programmer current is used to predict the current flowing through the agent. Using characterization curves such as the ones shown in Figure 35b (which account for the subthreshold slopes and the differences in current due to differing source potentials), the agent current can be accurately programmed. Using these characterization curves to set the programmer current that will yield the desired agent current, we show in Figure 35c that this technique can be used to accurately set the agent current within tolerance for two different values of the agent's source potential. While achieving high precision on the actual programmed current in the agent is important, the ultimate goal of accurate programming is to achieve precise control over the operation of the overall circuit, and this procedure will be described in detail in Section 5.3.5.

5.3.3 Indirect Programming of nFET Transistors

As stated previously, an important advantage of indirect programming is that it provides a simple mechanism for programming an nFET, whereas low injection efficiency makes direct nFET programming difficult. In this section, a pFET and an nFET share a common floating gate, as was shown in Figure 33b. Figure 36b shows the I-V characteristics of both the nFET and pFET. If the transistors are not properly sized, then the currents level at which both transistors have equal currents will be very high, as is illustrated in Figure 36c. Unlike the pFET-pFET case, a direct relationship between the two transistors is not easily obtained. When the two transistor currents are not in subthreshold simultaneously, a current-to-current relationship like that in curve 1 of Figure 36b is the result. Small changes in pFET current yield large changes in nFET current. Therefore, restricting the operation to strictly subthreshold is desirable because it linearizes the current-to-current ratio.

Two methods are available to ensure that both transistors are simultaneously in the subthreshold regime. The first method requires moving the sources of both

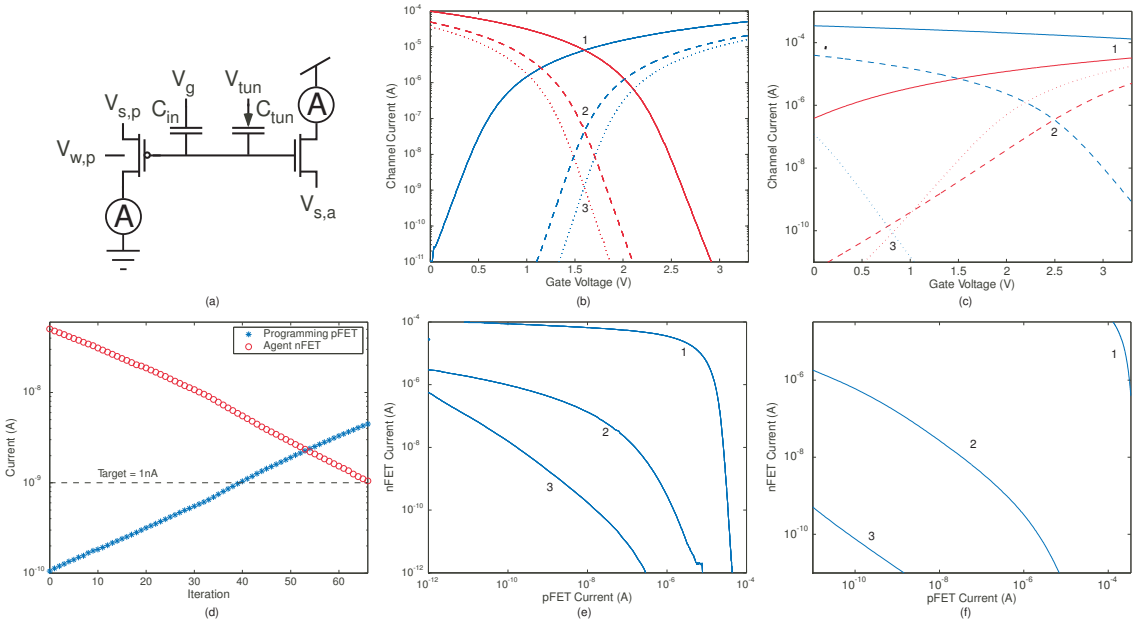


Figure 36. Indirect programming of an nFET transistor. (a) Testing setup for the following measurements using two identical picoammeters. These data are from an isolated nFET-pFET pair that has been used for characterization purposes. (b) I-V characteristics of an nFET-pFET pair ($\frac{W}{L} = 2$ for both). Curve 1 shows the I-V relationships when $V_{w,p} = V_{s,p} = V_{dd}$ and $V_{s,a} = gnd$. Curves 2 and 3 show the I-V relationships attained by increasing $V_{s,a}$ above gnd and by lowering $V_{w,p} = V_{s,p}$ below V_{dd}. Changing the source potentials of the nFET and pFET allow the two transistors to operate with subthreshold currents simultaneously, which is advantageous for accurately programming the nFET transistor. (c) I-V characteristics attained by raising $V_{w,p}$ and lowering $V_{s,p}$. The pFET is much (10 times) larger than the nFET. This is an exagerated example that would not typically be used but has been used here to illustrate the ability to achieve subthreshold operation in both transistors even in an undesirable case. Curve 1 shows the I-V relatioships when $V_{w,p} = V_{s,p} = V_{dd}$ and $V_{s,a} = gnd$. Both transistors have very large above threshold currents when they cross (if the gate voltage increased above V_{dd}) and require significant movements to both achieve subthreshold operation. Curves 2 and 3 show that by raising the pFET's well potential above V_{dd} and by lowing the pFET's source potential below V_{dd}, the operation of both transistors can be placed into the subthreshold regime. Even though the terminals of the nFET are not altered in this example, the current flowing through the nFET still changes. This change in current is due to capacitive coupling onto the FG node through the parasitic capacitances (C_{gs} and C_{gw}) of the pFET. This capacitive coupling, as well as the change in the subthreshold slope, will be explained in Section 5.3.4. (d) Programming the agent nFET to a target current within accuracy. Either the method of (b) or (c) can be used to place both transistors into subthreshold operation for programming. (e) Current-to-current relationships for each of the three curves shown in (b). (f) Current-to-current relationships for the method used in (c). By placing both transistors into subthreshold operation, a given percentage change the pFET's current translates into a linear percentage change in the nFET, therefore making the programming algorithm easier to implement and predict correct pFET current for the desired nFET current.

transistors. Decreasing the pFET source (reference to V_{well}) and increasing the nFET source (referenced to V_{bulk}) reduces the current in each transistor. This moves the threshold voltages to a point in which it is possible to operate both transistors in subthreshold at the same time (Figure 36a). Figure 36b relates the pFET-to-nFET current for each set of curves in Figure 36a. Lowering the crossover point increases the linear range of the current-to-current ratio.

A linear current-to-current relationship makes predicting the agent current trivial. However, any reasonable current-to-current relationship (like curve in 2 Figure 36b) allows accurate programming of the nFET.

As the source of the agent may not always be accessible or is set to a given potential due to placement within the circuit, the previous method is not always possible. The second method of ensuring that both transistors are in subthreshold requires that the programming pFET is in a well isolated from the operational circuit and that the well can be accessed. By raising the potential of the programmer's well and also lowering its source potential, the current flowing through the pFET is reduced. By using this procedure, the currents flowing through the nFET and pFET can be made to cross each other in the subthreshold regime. Figure 36c shows the operation of this process for the worst case scenario in which the pFET is much larger than the nFET $\left(\frac{W_p}{L_p} = 10\frac{W_n}{L_n}\right)$. Since the pFET is so much larger than the nFET, larger voltage differences from V_{dd} must be used in this example to bring the currents to be simultaneously in subthreshold operation. Typically, a nearly minimum-sized pFET programmer would be used, and the voltage differences would not be as large, but for illustrative purposes, we have shown that this operation is still possible under the worst-case scenario.

The movement of the nFET's current is due to capacitive coupling onto the floating gate, which will be explained in detail in the next section. Again, this movement is maximized in this example due to the large size of the pFET. Additionally, the change

in the subthreshold slope, as seen in Figure 36, is another result of the pFET's large size and would be minimized for smaller transistor sizes. This effect will also be discussed in the following section.

Either of these two methods can be used to accurately program a current in the nFET. By keeping both transistors in subthreshold and measuring the pFET's current, the linear relationship of either Figure 36e or Figure 36f can be used to predict the nFET's current. Figure 36d shows an example of accurately programming a current in the nFET where only the pFET's current is observable during the programming routine.

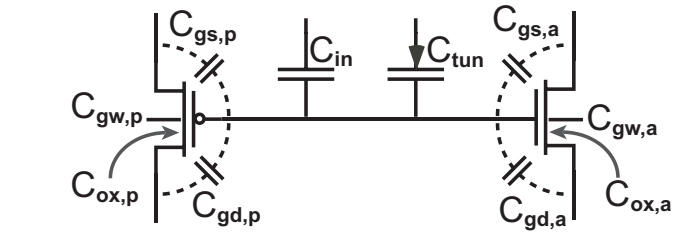
5.3.4 Capacitive Coupling with Indirect Programming

As has been shown previously, the difference between source potentials of the programming pFET and the agent transistor need to be taken into account when programming so that the correct current flows through the agent. The drain potentials of the two transistors are also of concern, especially the drain of the agent since the operation of its connected circuit can affect the potential at the drain. The terminals of the programming pFET is held constant when not programming, thus eliminating all transient coupling effects from it.

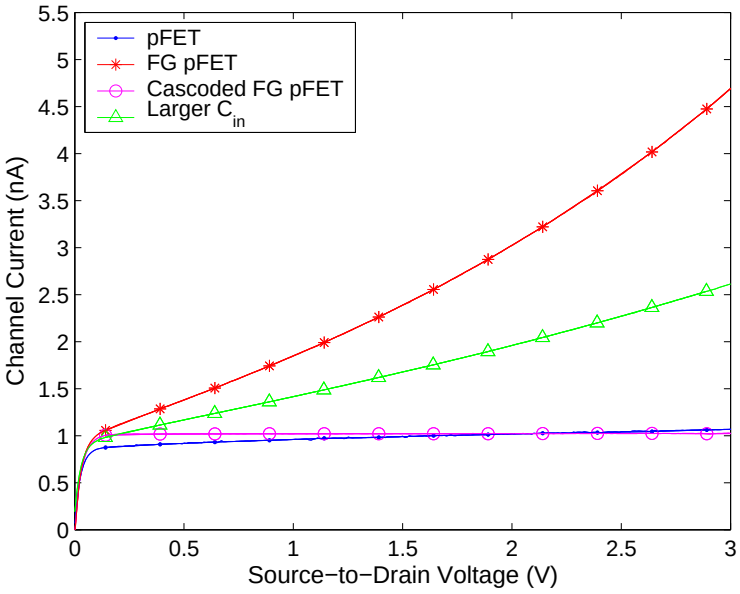
The voltage on any FG node is set by a combination of the FG charge and a sum of the inputs to the gate through capacitive dividers [57]. The extension of the the FG voltage for the indirect programming case is depicted in Figure 37a and described by

$$\begin{aligned} V_{FG} &= \frac{Q_{FG}}{C_T} + \frac{C_{in}}{C_T}V_g + \frac{C_{tun}}{C_T}V_{tun} \\ &+ \frac{C_{gd,p}}{C_T}V_{d,p} + \frac{C_{gs,p}}{C_T}V_{s,p} + \frac{C_{gw,p}}{C_T}V_{w,p} + \frac{C_{ox,p}}{C_T}\psi_p \\ &+ \frac{C_{gd,a}}{C_T}V_{d,a} + \frac{C_{gs,a}}{C_T}V_{s,a} + \frac{C_{gb,a}}{C_T}V_{b,a} + \frac{C_{ox,a}}{C_T}\psi_a \end{aligned} \tag{41}$$

where C_T is the total capacitance connected to the FG node, the p and a subscripts



(a)



(b)

Figure 37. (a) Schematic of a pair of transistors sharing the same floating gate and the parasitic capacitances that allow coupling of voltages onto the floating node. (b) Transistor drain sweeps. Due to capacitive coupling through $C_{gd,a}$, I_{sat} in the FG pFET increases exponentially for larger $V_{ds,a}$ values. Increasing C_{in} increases the effective Early voltage. Cascoding the agent transistor eliminates the exponential current increase and flattens I_{sat} more than the I_{sat} of the identically sized non-FG pFET.

indicate the programmer and the agent, and ψ represents the surface potential of each transistor (constant ψ in subthreshold). Since $C_{gd,a}$ is a small parasitic capacitance, the drain of the transistor acts as an input to the gate. As the drain voltage of the agent is swept, a subthreshold current through the device changes exponentially, as is shown in Figure 37b. This is a significant alteration from the small slope due to the Early voltage of an identically sized transistor, which is also shown.

In essence, this drain coupling of the agent can be viewed as reducing the effective Early voltage, which is undesirable if using the transistor as a current source. By rewriting (41) as

$$V_{FG} = \frac{C_{in}}{C_T} V_g + \frac{C_{gd,a}}{C_T} V_{d,a} + V_{offset} \tag{42}$$

where V_{offset} represents all the other terms in (41), replacing the gate term in the subthreshold equation (39) with (42), and dropping the a subscript for the agent, the saturation current becomes

$$I = I_0 e^{\kappa \left(\frac{C_{in}}{C_T} V_g + \frac{C_{gd}}{C_T} V_d + V_{offset} \right) / U_T} e^{-V_s/U_T} e^{V_d/V_A}. \tag{43}$$

Rearranging, this expression takes the form

$$\begin{aligned} I &= I_0 e^{\kappa V_{offset}/U_T} e^{\kappa \frac{C_{in}}{C_T} V_g / U_T} e^{-V_s/U_T} e^{V_d \left(\frac{1}{V_A} + \frac{\kappa C_{gd}}{C_T U_T} \right)} \\ &= I_0 e^{\kappa V_{offset}/U_T} e^{\kappa \frac{C_{in}}{C_T} V_g / U_T} e^{-V_s/U_T} e^{V_d \left(V_A \| \frac{C_T U_T}{\kappa C_{gd}} \right)}. \end{aligned} \tag{44}$$

The effective Early voltage is thus

$$V_{A,eff} = V_A \parallel \frac{C_T U_T}{\kappa C_{gd}}. \tag{45}$$

With typical capacitance values, the effective Early voltages for FG transistors can easily fall into the range of 1V, much like the the FG transistors shown in Figure 37b.

If supply headroom issues are important, then the drain-coupling effect can be minimized by increasing the input gate capacitance. Increasing C_{in} increases C_T, thereby reducing the effects of coupling through the parasitic capacitances, such as

$C_{gd,a}$. While the saturation current still has an exponential increase with drain potential, the effective Early voltage is increased, as is shown in Figure 37b.

If supply headroom issues are not a concern, then this drain-coupling effect can be completely removed by adding a cascode transistor at the drain of the agent. The saturation current received by the circuit is flatter than even a standard transistor, as is shown in Figure 37b.

Coupling through the gate-to-drain capacitances is not the only source of coupling into the floating node. In fact, all the terminals affect the drain currents of the two transistors to varying degrees by coupling into the floating node, as was shown in (41). These varying degrees depend on both the total capacacitance, C_T, connected to the FG and also the size of the capacitor through which the voltage couples, which is typically a small parasitic capacitance. Increasing C_T decreases the capacitive coupling affects, as does decreasing the parasitic capacitances through which the coupling takes place. For example, simply increasing the drawn C_{in} and using a minimum-sized transistor will reduce the effect of the overlap capacitance, C_{gd}, coupling into the floating gate.

For this reason, when programming a nFET-pFET pair and altering the pFET's source and well potentials, these voltages alter the charge on the floating node. This is the reason that the nFET's curve shifts in Figure 36c because the pFET is a large device, and the parasitic capacitances between the gate and source and the gate and well are comparable to C_T. For this reason, nearly minimum-sized programmer pFETs should be used when using an nFET-pFET pair to reduce the parasitic capacitances.

The second reason for making the programmer pFET small in an nFET-pFET pair is because of the change in the subthreshold slope when modifying the pFET's source and well potentials. The parasitic capacitances of a transistor are different depending on which mode of operation the transistor is in (subthreshold or above threshold). To minimize the changes in the coupling affects between modes of operation, the

transistors should be made small so that the input capacitance, C_{in}, dominates the total capacitance, C_T.

5.3.5 Precise Tuning of Circuits

If the DC operating point of the the agent transistor is not known, and the drain current has an exponential dependance upon all of its terminals, how can an indirectly programmed transistor accurately bias a circuit, especially in the case where no cascode is used to protect the drain terminal? Even though the current through the agent transistor is unobservable, the overall operation of the circuit can be tuned so precisely that the effects of device mismatches can be negated. In the following, we will give a circuit example that shows the method for programming both a pFET and an nFET for correct circuit operation. The C^4, which is a bandpass filter typically used in audio applications, serves as a useful example of indirect programming because the two corner frequencies are each set solely by the current flowing through a single transistor, where one must be an nFET and the other a pFET. Figure 38 shows the schematic of an indirectly programmed C^4.

5.3.5.1 pFET Programming

Programming an agent pFET to yield a desired circuit performance is a straightforward procedure. This process involves two steps in which a current is programmed into the pFET, and the effects of mismatch are then calibrated out.

The programmer is initially programmed to the current that should yield correct circuit performance if all devices were ideal. Using the designed values and a rubric for the correct circuit operation, an initial current is programmed into the programmer. However, all device parameters will deviate from the ideal, and since the DC operating point of the agent will likely differ from the programmer, the actual performance of the circuit will not equal the idealized performance. Nevertheless, once the programmed current and the resulting circuit performance are known, the function relating the

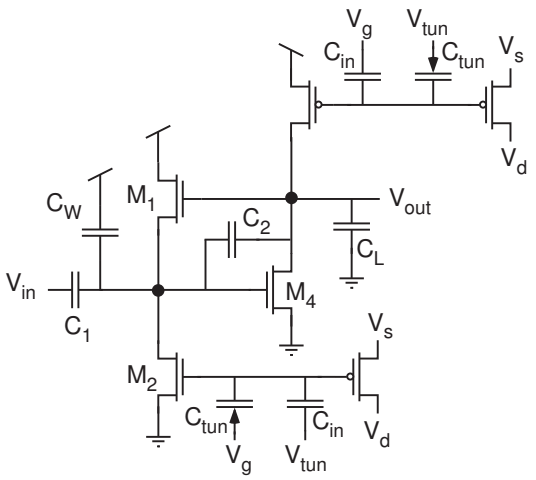


Figure 38. Schematic of an indirectly programmed version of the capacitively coupled current conveyor (C^4). This bandpass filter is used as an example of programming a circuit to a desired performance because the two corner frequencies are each tuned solely by altering the current through a given transistor. The high corner frequency is tuned by programming the current through a pFET ($I_{\tau h}$). The low corner frequency is tuned by programming the current through an nFET ($I_{\tau l}$). Thus the C^4 is a good example for showing the operation of programming both a pFET and an nFET indirectly for a desired circuit performance.

two can be calculated. This function incorporates both the deviations from the ideal device parameters and also the difference in DC operating points of the programmer and agent, and, thus, the circuit can be reprogrammed to any desired performance.

Using the example of the C^4, the pFET agent exclusively controls the high corner frequency. The rubric for knowing correct circuit operation is thus the placement of the high corner frequency, which is given by

$$f_h = \frac{1}{2\pi} \frac{C_2}{C_T C_O - C_2^2} \frac{\kappa_{p,eff} I_{\tau_h}}{U_T}. \tag{46}$$

where $\kappa_{p,eff}$ is the effective coupling onto the surface potential including the input capacitor, C_{in}. An initial current is programmed into the the programmer assuming ideal values for the capacitors and $\kappa_{p,eff}$ such that the resulting corner frequency should be the target value. Since these idealized values are not the actual values, and since the drain of the agent is not the same as that of the programmer, the actual programmed corner frequency does not fall within tolerance of the target value.

However, the function relating the corner frequency and the current only involves a single coefficient since the currents are remaining in subthreshold and (46) applies.

Equating all the coefficients of the programmed current into a single coefficient, (46) becomes

$$f_h = K_{high} I_{\tau_h}. \tag{47}$$

Since the programmed current and the circuit output, the corner frequency, are known, the true value for K_{high} can be calculated. Now,

$$K_{high} = \frac{1}{2\pi} \frac{C_2}{C_T C_O - C_2^2} \frac{\kappa_{p,eff}}{U_T} k_{DC} \tag{48}$$

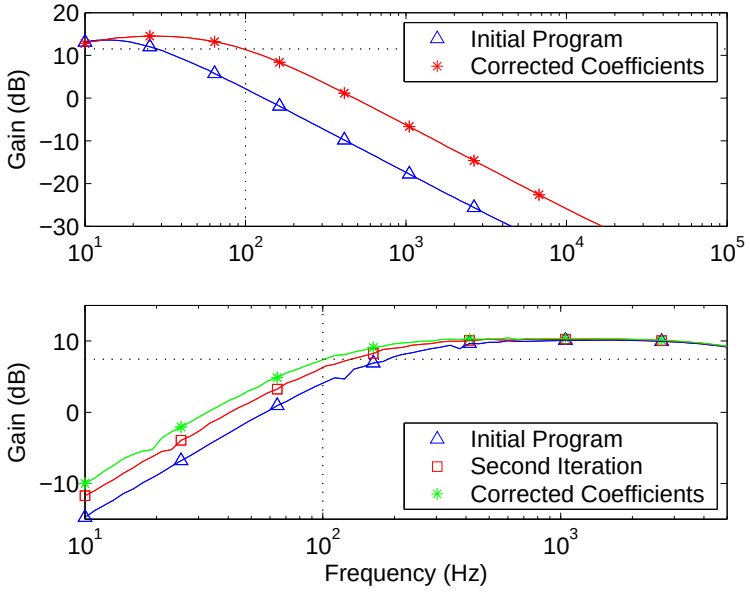
where all the device parameters represent their actual values, and k_{DC} represents the shift in the bias current between the agent and the programmer due to differences in the DC operating point. Using (47) with the measured value of K_{high} allows a second programming step to be used to produce the desired corner frequency.

Figure 39a shows an example programming the C[4]'s high corner frequency using this method, and the high-degree of accuracy with this approach is clear. The crosshairs indicate the location of the ideal -3dB frequency. Further improvements in accuracy can be achieved by improving the accuracy of the FG programming algorithm, as is described in [43].

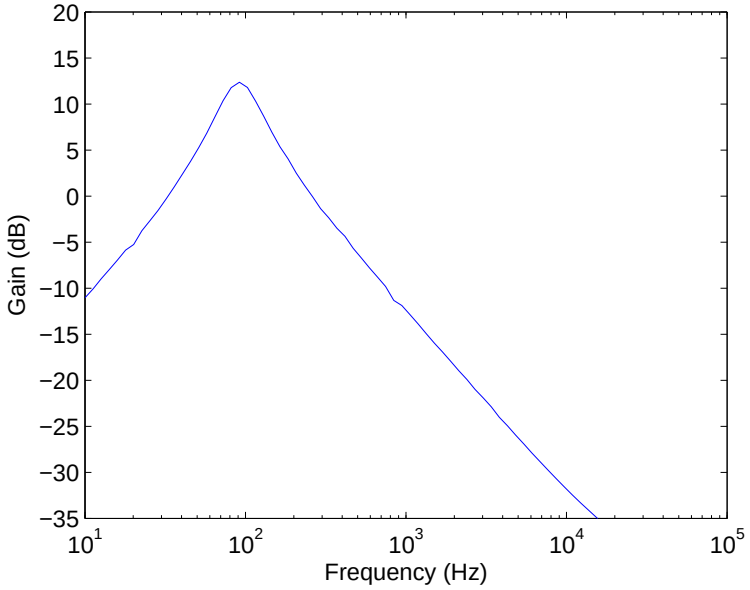
5.3.5.2 nFET Programming

Since the current through an nFET agent follows an inverse relationship to the current through its pFET programmer, programming a precise current is a more complicated procedure than a pFET-pFET case. A high degree of characterization of the nFET-pFET combination will ease the programming procedure. However, this characterization is not required, and through the following example, we will show how to achieve accuracy even when an exact relationship between the nFET and pFET is not initially known.

The procedure starts by programming an initial current into the programmer that



(a)



(b)

Figure 39. (a) Indirectly programming both corner frequencies of a C^4 to a target of 100Hz. (top) Programming the high corner frequency , or the current through a pFET, requires only two steps to achieved a desired value within tolerance. The crosshairs show the location (frequency and gain) of the desired -3dB frequency. As can be seen, the actual -3dB frequency matches the target well within the allotted tolerance. This programming technique essentially eliminates the effects of mismatch of device parameters. (bottom) Programming the low corner frequency requires three steps. The additional step is used to determine an exact relationship between the programmer current and the agent current. (b) By programming both corner frequencies of the C^4, to the desired value, the circuit takes on the form of a bandpass filter.

will translate as closely as possible to an nFET agent current that will yield the desired circuit operation. The translation from programmer current to agent current can be estimated by a characterization nFET-pFET pair on the periphery of the die area or even by simulation. A circuit measurement is taken to determine the deviation from the ideal performance. This difference will be due to deviations in parameter sizes and values as well as differences in the agent current from the expected value.

Whereas simply finding the estimate of the device and current mismatch for a given parameter was sufficient for the pFET-pFET case, this method is no longer sufficient for the nFET-pFET case. Placing both the programmer and agent transistors into subthreshold simultaneously greatly eases the programming procedure since the relationship is linearized (on a logarithmic scale), as is shown in Figure 36e-f. Two calibration steps are required to accurately program an nFET. The first calibration step allows the effective mismatch to be found, and the second step allows the slope of the relationship between the programmer and the agent to be determined. Then the current can be programmed so that the circuit accurately performs the desired action.

Again, we will use the C^4 as a circuit example, and since the low corner frequency of the C^4 requires only the current through the nFET agent to be modified, the low corner frequency will be the system parameter under study. The low corner frequency is given by

$$f_l = \frac{1}{2\pi}\frac{\kappa_{n,eff} I_{\tau l}}{C_2 U_T}. \tag{49}$$

Using an estimate for the nFET agent's current and the ideal values for the device parameters, a current is programmed such that the low corner frequency should hit its target. However, the actual corner frequency will likely deviate from the desired value due to both device mismatch and the difference from the desired nFET current. The actual corner frequency will have a value of

$$f_1 = \frac{1}{2\pi}\frac{\kappa_{n,eff} I_{n1}}{C_2 U_T} = K_{low} I_{n1} \tag{50}$$

where f_1 is the initial measured corner frequency, K_{low} is the estimated multiplicative coefficient, and I_{n1} is the unknown and unobservable agent current.

In addition to the unknown agent current, the *relationship* between the programmer and agent currents is also not yet known. An alternative way of viewing this problem is that the slope of the curve in Figure 36f is not known, even when assuming subthreshold operation. A second current must be programmed into the programmer using (50) such that

$$f_2 = K_{low} I_{n2} \tag{51}$$

where I_{n2} is the agent current and f_2 is the resulting corner frequency. However, this new corner frequency will likely not fall within tolerance because the exact value of I_{n2} is unobservable.

Nevertheless, there is now enough information to program the circuit accurately on a third iteration, and this is done by finding the slope of Figure 36f, assuming subthreshold operation. To find this slope, (50) is divided through by (51).

$$\frac{f_1}{f_2} = \frac{K_{low} I_{n1}}{K_{low} I_{n2}} = \frac{I_{n1}}{I_{n2}} \tag{52}$$

Then, letting m represent the slope of Figure 36f and using the ratios of (52), the slope is given by

$$m = \frac{\ln(I_{n2}) - \ln(I_{n1})}{\ln(I_{p2}) - \ln(I_{p1})} = \frac{\ln\left(\frac{I_{n2}}{I_{n1}}\right)}{\ln\left(\frac{I_{p2}}{I_{p1}}\right)} = \frac{\ln\left(\frac{f_2}{f_1}\right)}{\ln\left(\frac{I_{p2}}{I_{p1}}\right)} \tag{53}$$

Now, letting k represent the programming iteration number, the slope can be written as

$$m = \frac{\ln\left(\frac{f_{k+1}}{f_k}\right)}{\ln\left(\frac{I_{p,k+1}}{I_{p,k}}\right)} \tag{54}$$

where knowlege of only the programmer currents and resulting corner frequencies are required. Rewriting (54) and letting f_{k+1} represent the desired corner frequency, the exact current that must be programmed into the programmer is given by

$$I_{p,k+1} = I_{p,k} \left(\frac{f_{k+1}}{f_k}\right)^{1/m}. \tag{55}$$

Thus, in three steps, the relationship between the nFET and pFET has been determined, the effects of mismatch have been calibrated out, and the circuit has been programmed to the desired corner frequency.

Figure 39a shows data from this programming procedure for the C^4's low corner frequency. On the third iteration, the corner frequency fell well within the tolerance of the programming algorithm, as is indicated with the ideal -3dB point depicted with the crosshairs. Again, this percentage error could be improved even further by increasing the accuracy of the programming algorithm. Figure 39b shows the response of the C^4 after both corner frequencies have been programmed.

5.3.5.3 Generalized Indirect Programming Algorithm

While the C^4 served as a good example of indirectly programming a circuit for precise operation criteria, the C^4 is by no means an exclusive case. In fact, this indirect programming algorithm can be applied to a wide variety of circuits, and it can be viewed in its generalized form to be as that described in the flow diagram of Figure 40.

In all cases, the circuit should be initially programmed so that it would perform perfectly if all device parameters were ideal. Some circuit measurement should then be taken, be it a frequency response, a step response, *etc.*, to determine how far from ideal the circuit's performance was. If at any time, this circuit operates within system tolerance, then no more steps are needed. However, the initial program will not likely produce the desired results, but it can be used to extract certain parameters about the circuit's operation. These parameters may include a variety of contributions, including capacitor sizes and transistor currents. These extracted parameters can then be used to reprogram the circuit, and the circuit is then tested again to find whether or not it operates within the desired tolerances.

This loop, as shown in Figure 40, can be repeated as many times as necessary. Typically, only a single time through the loop is required for programming a pFET since both the programmer and the agent follow the same current trends. A second

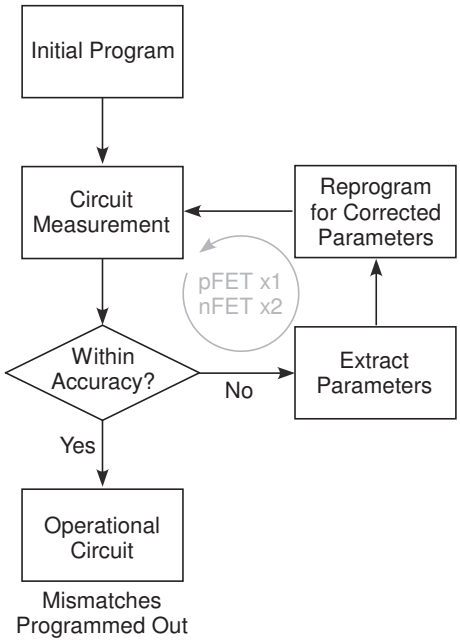


Figure 40. Flow diagram of the programming algorithm used in tuning a circuit to the desired performance. Programming a pFET indirectly requires a single iteration through the loop, whereas programming an nFET indirectly requires two iterations through the loop. The additional iteration for the nFET results from the need to determine an exact relationship between the programmer pFET current and the agent nFET current.

iteration through the loop is required for indirectly programming nFETs since not only device parameters must be determined but also the exact relationship between the currents in the nFET agent and the pFET programmer. Keeping both the agent and the programmer in subthreshold simultaneously aids this procedure since the relationship is linearized, as is shown in Figure 36e-f.

5.3.6 Benefits of Indirect Programming

In addition to the ability to program out mismatches in a circuit and set precise current sources, which are both advantages available with FG circuits and direct programming methods, the non-invasive nature of indirect programming has several benefits over traditional FG programming methods. These benefits are largely related to the removal of the transmission gates that are needed for disconnecting FG transistors for a programming phase.

The addition of at least one T-gate for every FG transistor, and often more T-gates for certain circuit configurations [54], adds both resistance and capacitance to the FG circuit. The added resistance and capacitance can have several harmful effects. These extra parasitics will slow down the operation of the circuit and, thus, limit the speed at which the circuit can operate. Also, when using large currents in the FG transistors, the added resistance, which is approximately $10k\Omega$ for small devices [58], will cause a significant voltage drop to form across the switch. The voltage drop could cause problems with the operation of the circuit and it could be large enough to alter the required voltage headroom of the circuit. Thus the circuit would have to run on a larger supply voltage.

Since indirect programming of FG transistors does not require this disconnection via T-gate switches, many of the parasitics are removed. Therefore, IPFG circuits have the ability to operate at higher frequencies than do directly programmed FG circuits. The increase in speed with IPFG transistors was demonstrated with an IPFG inverter using an ad hoc programming method [59]. Moreover, this IPFG inverter was able to operate at faster speeds than an identically sized non-FG inverter. Furthermore, the removal of the selection switches removes the added resistance. Circuit applications requiring very low supply voltages can now utilize the programmability of FG transistors without concerns of headroom loss due to parasitic resistances.

5.3.7 Run-Time Programming

Since the use of indirect programming does not require disconnection of the agent transistor from its circuit, there is now no need for a separate programming phase to set the charge on the floating-gate nodes. In fact, programming can occur during normal operation of the circuit so that data acquisition does not need to be stopped in order to reprogram the device. This new "run-time programming," which will be introduced in this section, allows a circuit to be recalibrated while it is still operating so that the circuit can respond to changes in its environment (e.g. temperature) or

new desires of the circuit's user (e.g. changing the gain of a certain band of frequencies in a hearing aid). This run-time programming, unlike adaptation techniques, allows programming to be turned on temporarily whenever recalibratin is desired.

While typical methods of programming floating-gate transistors [53] would work even in this run-time programming, these methods are not ideal since they involve large, instantaneous movements of the transistor's terminal voltages in order to cause injection to occur. Since, with indirect programming, the programmer and the agent share the same FG node and the movements on the programmer's terminals capacitively couple onto the FG node, these methods of programming can cause large instantaneous changes in the agent's current that could seriously alter the operation of the circuit. Therefore, when recalibrating a circuit while it is still operating, care must be taken so that the operation of the circuit will not be temporarily rendered useless (and thus negating the benefits of using of run-time programming).

To recalibrate an FG agent in run-time operation using injection, the actual charge on the floating node should remain unaltered by any process except for injection. Therefore, any voltages that couple onto the floating node should always be balanced by an equal voltage coupling onto the floating node in the opposite direction. Referring back to (41) and Figure 37a, if one terminal of the programmer is moved, then another terminal must also be moved in the opposite direction such that the two voltages couple identical, but opposite amounts. The current flowing through the agent will thus not be moved at all. By pulling the source and drain apart "symmetrically" about V_{FG}, the source-to-drain potential is increased until the point at which injection occurs. When injection occurs, the charge on the floating node is altered, and the current flowing through the agent is modified (increased for a pFET and decreased for an nFET). When the current flowing through the agent has reached the desired value, then the injection can be turned off by returning the source and drain potentials to their normal operating values. Figure 41 shows the operation of

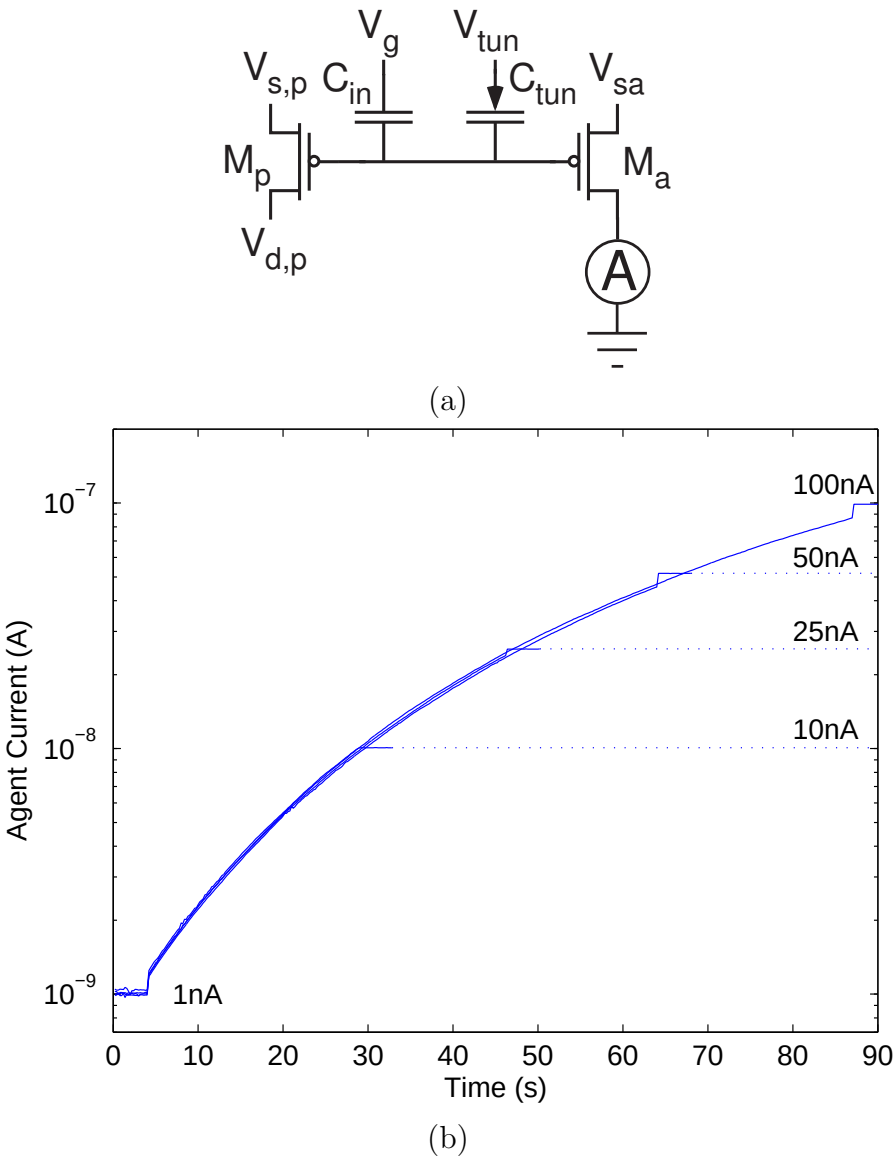


Figure 41. Run-time programming using indirectly programmed floating-gate transistors. (a) Schematic for programming the agent current using run-time programming. (b) At $t = 5$s, injection was turned on by symmetrically changing the source, well, and drain potentials of the programmer such that the contribution of all coupling terms negated each other and the FG voltage remainded stationary. Once injection started, electrons were added to the FG, and the agent current started to increase. Injection was turned off (all the programmer terminals were symmetrically brought back to their initial position) when the agent current reached its target value. The curvature to the slope shows that the injection efficiency decreases as the currents near threshold operation. Programming speeds can be increased to the microsecond timescale by increasing the source-to-drain potentials.

injecting the current to the desired value with this process.

The small discontinuities in the current levels at the onset and termination of injection are a result of parasitic-capacitance estimates not being perfectly calibrated. Additionally, the larger jump at the termination of injection is a result of the higher current levels (near or above threshold) and the resulting changes in capacitor values since the parasitic capacitances have different values when the transistor is operating in either subthreshold or above threshold. These discontinuities can be accounted for, and, thus, injection can be turned off in anticipation that the final current will be the desired value. These discontinuities can also be calibrated out and compensated in a manner similar to [60].

To test the operation of run-time programming within a circuit, the circuit of Figure 42a was built to show that by viewing the output of the circuit, the operation of the circuit can be recalibrated by using run-time programming. This circuit is simply a G_m-C element constructed to act as a first-order lowpass filter in which the time constant is set by an indirectly programmed transistor. The G_m element is simply a five-transistor operational transconductance amplifier (OTA) [1] in which the bias current is set with an IPFG transistor.

In this simple experiment to show how run-time programming can be used, the corner frequency of the filter was programmed to below 10Hz. However, it was desired that this corner frequency should be moved to exactly 400Hz without stopping the operation of the circuit. As a result, the output of the filter was viewed when injection was turned on in the programmer pFET. Injection was then turned off when the circuit was observed to be operating at the desired corner frequency. Figure 42b shows frequency responses before and after the run-time programming as well as the observed output of the circuit while injection was occurring. The final output of the filter had the desired corner frequency.

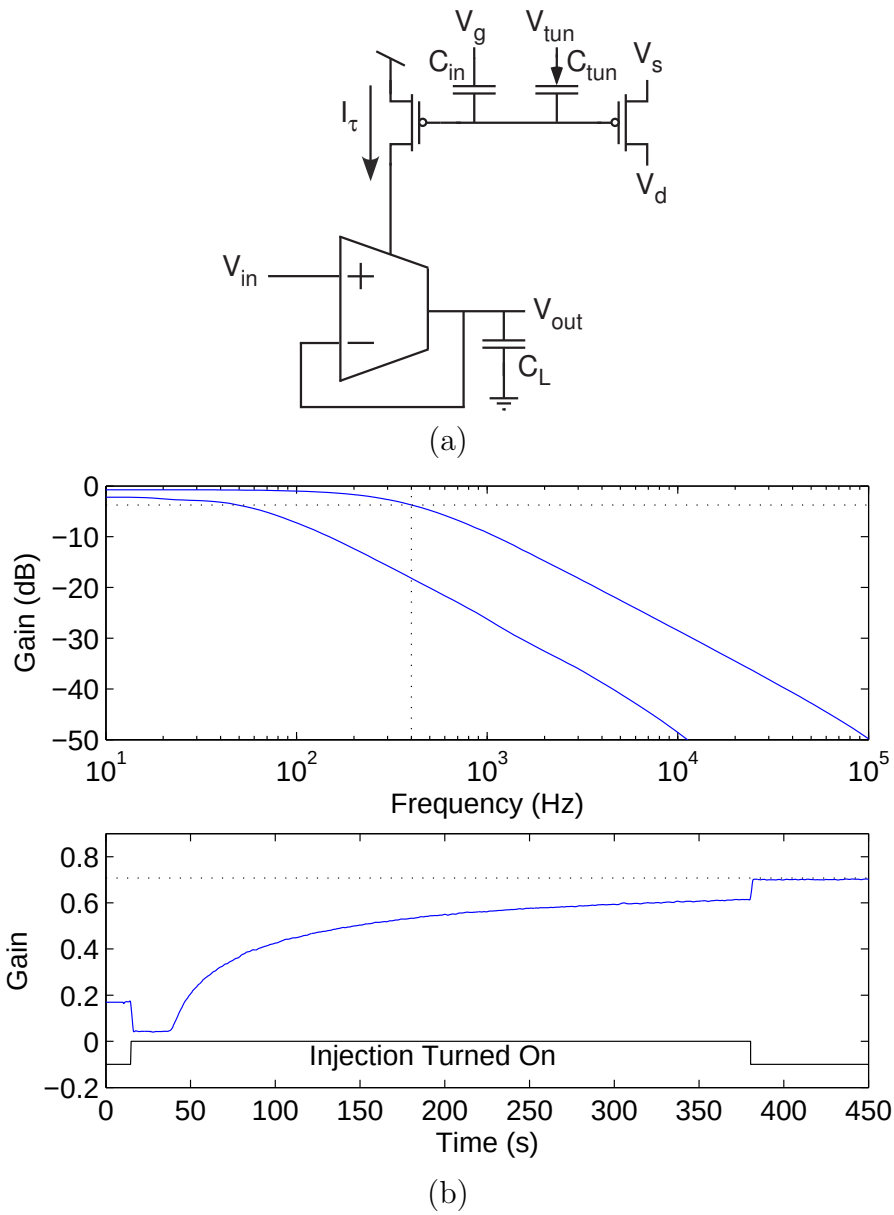


Figure 42. Run-time programming of a lowpass filter. (a) Simple G_m-C first-order lowpass filter using an indirectly programmed tail current. (b) The filter initially had a corner frequency below 10Hz and was to be reprogrammed to 400Hz without stopping the operation of the filter. By looking at the output of the filter for an input of a 400Hz sinusoidal waveform, injection was turned on and then turned off again when the amplitude of the filter reached the desired value. (Top) The frequency responses at the beginning and end of the run-time programming. The crosshairs show that the -3dB point is at the target frequency. (Bottom) The output of the filter while the filter was actively being programmed. The small change in amplitude at the onset and termination of injection was due to slightly unsymmetric coupling onto the FG node. A large current was required for these frequencies due to the size of the load capacitance. The slightly unsymmetric coupling was used because, at the large currents required, the injection efficiency was very low, and the unsymmetric coupling allowed a more efficient current level to be used.

This run-time approach to programming FG transistors has promising new possibilities for circuits needing frequent updates due to environmental changes and consumer needs. Additionally, a circuit using a similar approach has been used for adaptive applications by continuously updating the stored charge on the FG node [61].

5.4 Summary of Floating-Gate Transistors

Direct programming of floating gates has been proven to be a highly accurate tool for analog designers. Difficulties with programming nFETs and the parasitics associated with the isolation circuitry exist with the direct programming method but can be overcome by the indirect programming method we have just introduced.

An early, ad hoc method of indirectly programming floating gate transistors has been shown to be a useful means of tuning a circuit [62]. Additionally, we have presented a systemmatic approach to programming both pFETs and nFETs indirectly. This systemmatic approach can easily be extended to large arrays of FG devices so that a large number of current sources can be programmed without invasively disconnecting them. In fact, directly and indirectly programmed FG transistors can coexist in the same large array so that each might be used to its own particular advantage.

Indirect programming also allows certain circuits to be transformed into a programmable version that would not have been previously possible. The aforementioned programmable nFET current mirror is now possible, and a neuron circuit [63] that cannot properly operate due to the parasitics of the isolation circuitry can now be made.

New possibilities with floating-gate programming also exist. Since the agent transistor is never removed from its circuit, indirect programming removes the necessity of a separate programming phase and a operational phase. This allows for the possibility of run-time recalibration and adaption to be carried out by the programming

pFET.

Indirect programming offers solutions to many of the problems of direct programming while also providing new and unique capabilities to augment the analog designer's toolbox.

CHAPTER 6
A PROGRAMMABLE ARRAY OF BANDPASS FILTERS

In order to improve upon the accuracy of the time constants of the array of bandpass filters used in the cochlea model and to also provide programmability to the filters, floating-gate (FG) transistors were introduced into the circuit design. The resulting schematic of the C^4s is shown in Figure 43. The FG transistors are used in this application as precise, programmable current sources to set the corner frequencies of the bandpass filters. While either direct or indirect programming methods could be employed for accurate programming, and integrated circuits of both varieties have been fabricated, direct programming was found to be more adventageous for this particular design since a ratioed current mirror could be used to scale currents to the very low levels required for the low-frequency time constants.

An array of 32 FG programmable differential C^4s was fabricated in a 0.5μm process available through MOSIS. Using inspiration from biology, the center frequencies were programmed to have exponential spacing with narrow bandwidths and moderate amounts of resonance.

6.1 Exponentially Spaced Currents

To achieve exponential spacing of the bandpass filter center frequencies, the bias currents were programmed to be exponentially spaced (to within 95% accuracy). This is a similar situation to the resistively biased filter bank that achieved roughly exponential spacing in the bias currents through the use of a resistive divider. However, this programmable version allows the currents to be set precisely and directly by programming the FG transistor currents. This programmable version also has the added benefit of versatility in that the spacing and bandwidths of each filter may be altered at any time by reprogramming.

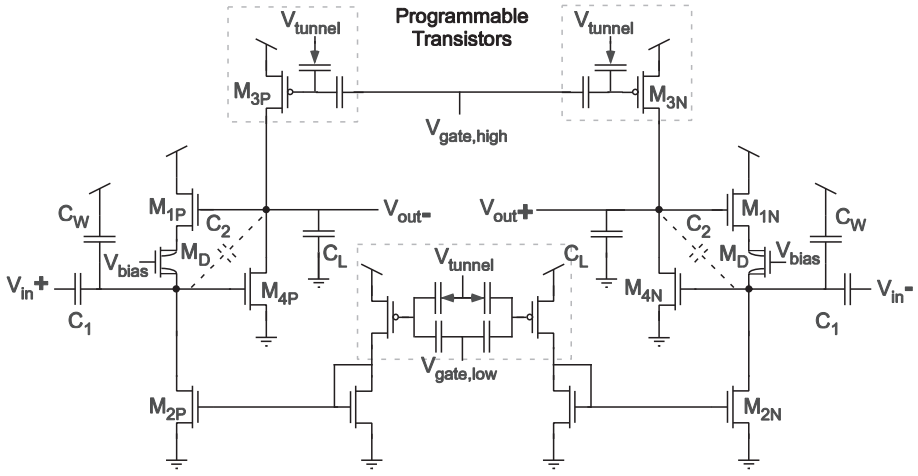
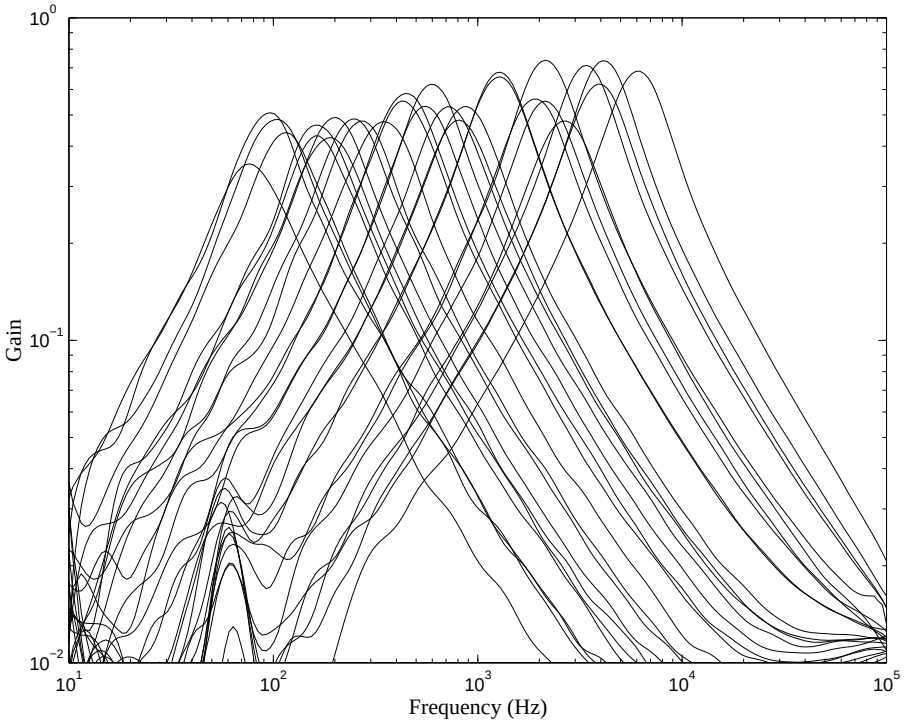


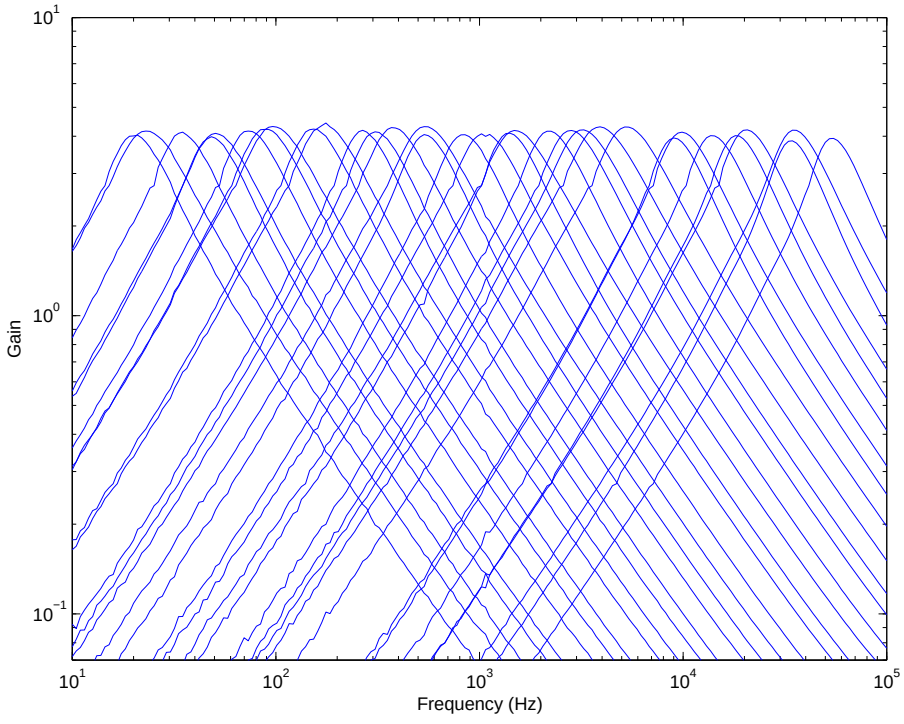
Figure 43. Differential, programmable C^4 used in the filter bank. Directly programmed C^4s were used instead of indirectly programmed versions because present methods of FG programming require off-chip current measurements. The currents used in the C^4 are smaller than can accurately be measured. Using ratioed current mirrors allows the small currents to be gained up to a value that can be measured.

Figure 44b shows the frequency responses of each filter tap, and the response of the resistively biased filter bank from Chapter 4 is repeated in Figure 44a for comparison. As can be clearly seen, the programmed filter bank is much more neatly tuned than the non-programmable version. The 32 traces from the programmable bandpass array are monotonically spaced, which is a difficult task and has only been overcome in the past on a consistent basis by using layout matching techniques, larger devices, and clever use of parasitic BJTs in standard CMOS processes [12]. In contrast, this monotonicity and spacing was simply programmed by the floating-gate biases.

However, careful inspection of Figure 44b shows that corner frequencies are not *perfectly* spaced, and these responses illustrate a fundamental design issue with analog circuits. No matter how accurately biases can be set, circuit performance is affected by mismatches that occur during the fabrication process. This is of no concern, though, because these errors due to mismatch of transistor and capacitor sizes can be simply programmed out with the floating gates.



(a)



(b)

Figure 44. Array of 32 programmable C^4s (a) Original array [23] using large resistive line to bias the transistors. (b) New array in which the time constants are set by floating-gate transistors that were programmed with exponentially spaced corner frequencies within 95% accuracy. This programmed array shows a marked improvement over the original, non-programmable array.

6.2 Programming to Remove the Effects of Device Mismatch

To alleviate the problems of center frequency accuracy, the bias currents must not be perfectly exponential in nature, but they should slightly deviate from the theoretical values to compensate for the effects of device mismatch. To determine the exact bias current that should be programmed into each floating gate of a C^4, a measure of the total effective mismatch for each time constant must be obtained.

The process of calculating and using this correction factor is as follows and is depicted visually in Figure 45a. Using the time constant equations (2) and ideal values for the the device constants and capacitances, an initial current is programmed into each bias FG transistor such that the upper and lower corner frequencies are widely spread and do not influence each other. The magnitude of the frequency response for the filter is then measured. The time constants can then be easily extracted by transforming these data and performing a linear regression. The transformation for highpass and lowpass responses (and thus low and high corners, respectively) can be written as follows:

$$y_L(x = \frac{1}{\omega^2}) = \frac{1}{|H(jw)|^2} = \frac{1}{A^2} + \frac{1}{\tau_L^2 A^2} \;\; x \tag{56}$$

$$y_H(x = \omega^2) = \frac{1}{|H(jw)|^2} = \frac{1}{A^2} + \frac{\tau_H^2}{A^2} \;\; x \tag{57}$$

where A is the passband gain. Using the extracted frequency response parameters, a correction constant is calculated as a ratio of the actual time constant and the target time constant. This correction factor is then multiplied with the original target current and the floating-gate devices are reprogrammed.

Viewed in another manner, the two corner frequencies are programmed using ideal device parameters such that the corner frequencies should hit a desired target value. However, due to device mismatch, the time constants will likely fall outside the allowed tolerance. Knowing both the measured -3dB frequency and the programmed current allows the exact function relating the two to be computed, which can be represented

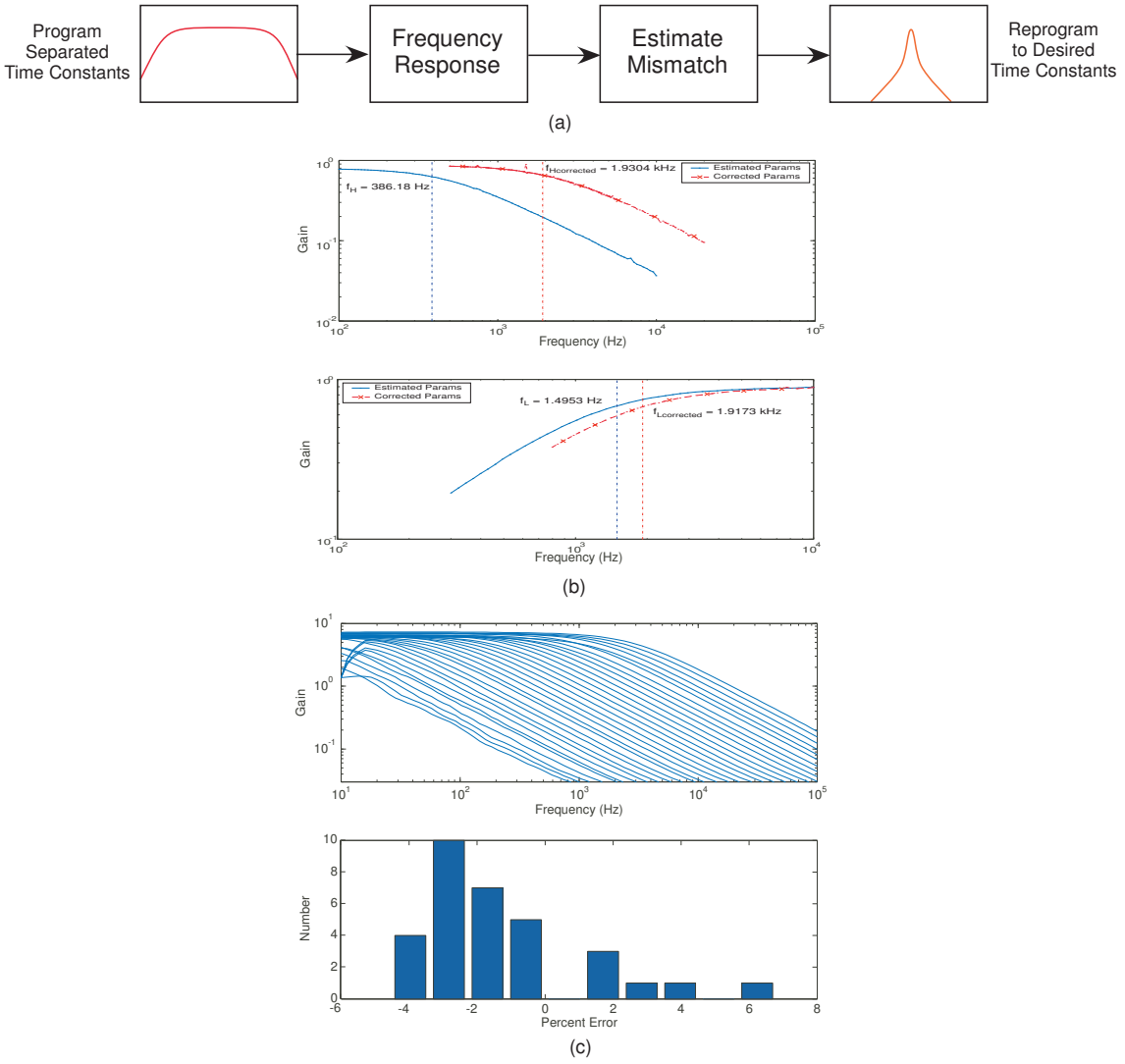


Figure 45. Programming out mismatches with floating-gate transistors. (a) Flow chart for programming out mismatches within a C^4. (b) Calibrating out the effects of mismatch for the two corner frequencies. A correction factor is calculated as the ratio of the actual time constant to the desired time constant and is used in determining the correct bias current to program. In the case of the high-corner frequency (top), the target corner frequency was 2.0 kHz; after one correction step, the corner frequency was measured as 1.93 kHz, which was a 4% error which was within the 5% tolerance characteristic of the programming algorithm. (c) Programming the high corner frequencies for the array. After calibrating the effective mismatch for the time constant for each filter tap, the high corner frequencies were programmed into the array with a high degree of accuracy (within $\pm 5\%$). Improving the measurement of both the FG transistor current and also the output of the C^4 would allow an even higher degree of accuracy to be reached.

by

$$\tau_h = \frac{K_h}{I_{\tau h}} \qquad \rightarrow \qquad K_h = \frac{C_T C_O - C_2^2}{C_2} \frac{U_T}{\kappa} \tag{58}$$

$$\tau_l = \frac{K_l}{I_{\tau l}} \qquad \rightarrow \qquad K_l = \frac{C_2 U_T}{\kappa} \tag{59}$$

Both K_l and K_h represent the *exact* values of the device paramters, including the effects of mismatch. Therefore, knowing these two coefficients allows the correct current to be determined for any desired time constant. As a result, two programming steps and one measurement step permit the time constants to be set *precisely*.

This method of estimation and correction has proven very successful. Data from a programmable C^4 circuit are shown in Figure 45b [64, 65]. The plots show both the original and the corrected frequency response curves of the low and high frequency corners. While the accuracy of the programmed corner frequencies was low when using the ideal values, after the correction factor had been applied, the new corner frequencies were within the 5% tolerance allowed for in the programming algorithm. The level of accuracy in this tuning algorithm is due primarily to the accuracy of measuring both the programmed current and the resulting corner frequency. Increasing the accuracy of these two measurements would allow any desired accuracy to be achieved.

Reprogramming the array with the adjusted currents yields a very high degree of accuracy. Figure 45c shows the result of programming all the high corner frequencies to their desired values after the calibration step. These programmed corner frequencies were within the 5% tolerance allowed with the programming algorithm.

Figure 46c shows the result of programming the entire array (both time constants) to their desired values. As can be seen, this is an improvement over programming the array with simply exponentially spaced currents. Figure 47 shows that the spacing of the individual center frequencies is monotonic and very evenly spaced. At very low frequencies, the current levels are very low (pA to fA currents), so that

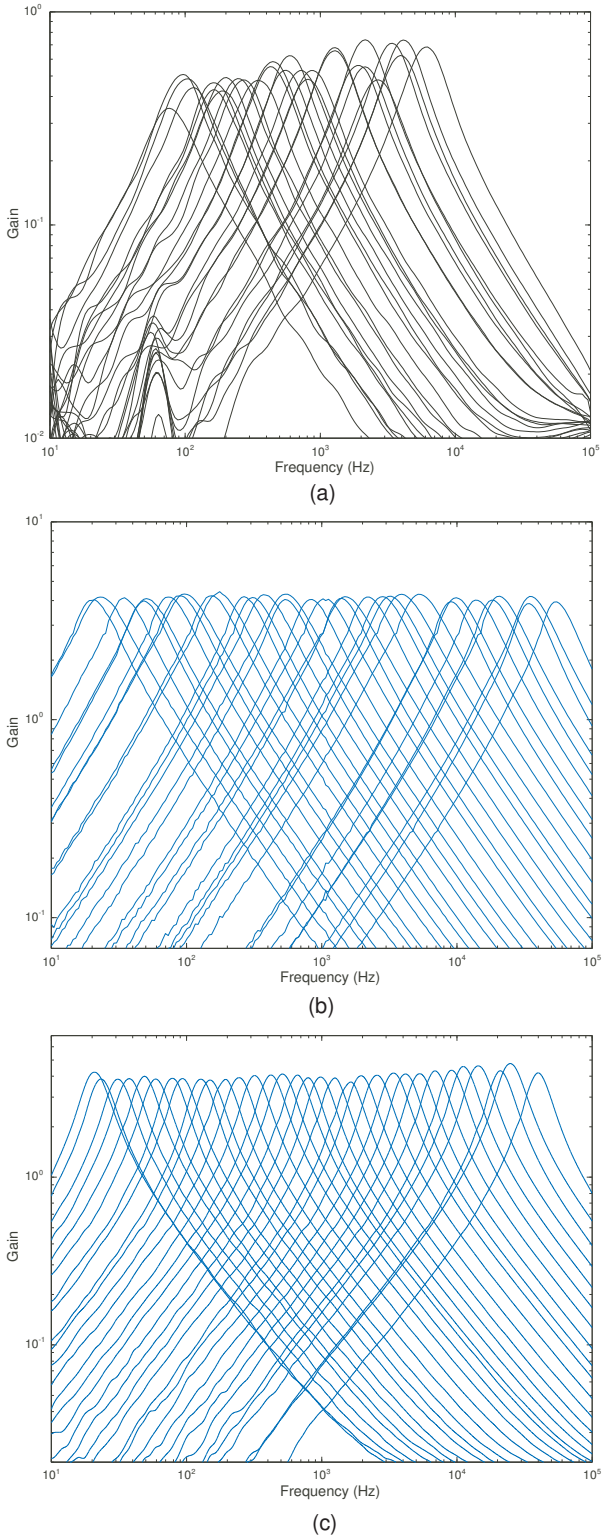


Figure 46. Programming the filter bank to remove the effects of mismatch. (a) Frequency response of the entire array of resistively biased C^4s. (b) Frequency response of the entire array of C^4s programmed with exponentially spaced currents. (c) Frequency response of the entire array of C^4s programmed to account for device mismatch. As can be seen, this is an improvement over programming exponentially spaced currents.

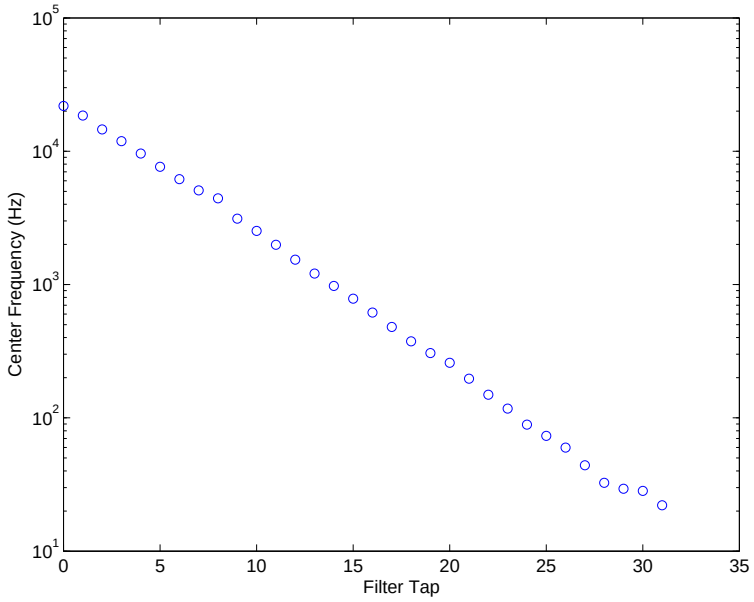
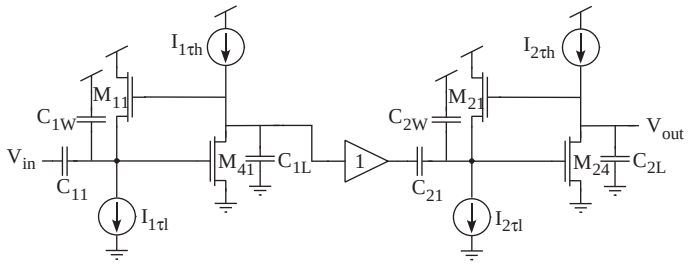


Figure 47. Center-frequency spacing of the programmed filter bank. The resulting center frequencies of the individual filter taps are monotonic and evenly spaced. This spacing is a marked improvement over the resistively tuned filter bank of Chapter 4 and of the cascade approaches of modeling the human cochlea [11, 33].

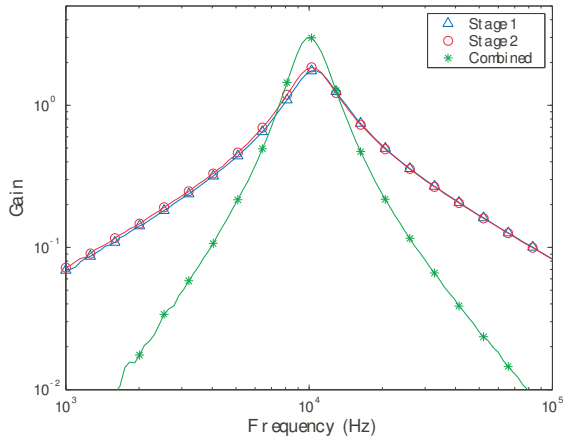
precise measurement is very difficult with off-chip measurement techniques. On-chip measurement techniques for programming floating-gate transistors is currently being developed and will allow much more accurate measurements of these current levels.

6.3 Second-Order Sections

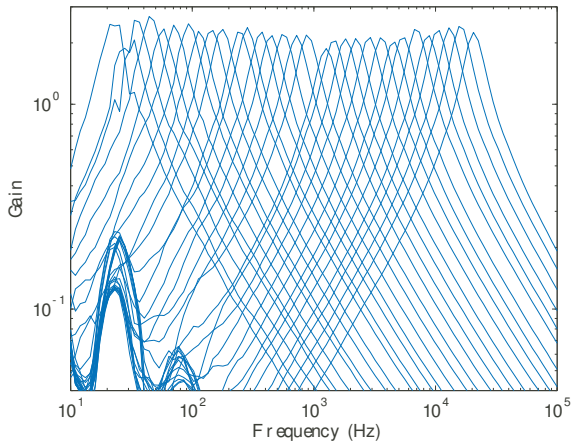
To achieve second-order slopes in the stopband and large levels of resonance, the C^4s can be cascaded, as is shown in Figure 48a. This configuration is referred to a C^4 SOS in reference to the early second-order section filters and because of its second-order slopes on both sides of the center frequency. A buffer is placed between the two C^4 stages so that shifting input impedance with varying frequencies does not adversely affect the filter [32]. Additionally, the buffer provides a convenient place to include transmission gates to select the first- or second-stage C^4 for calibration. The time constants of the two C^4s must be closely aligned if the center frequency and Q are to be accurately set. Therefore, the accuracy of the programming of each stage is



(a)



(b)



(c)

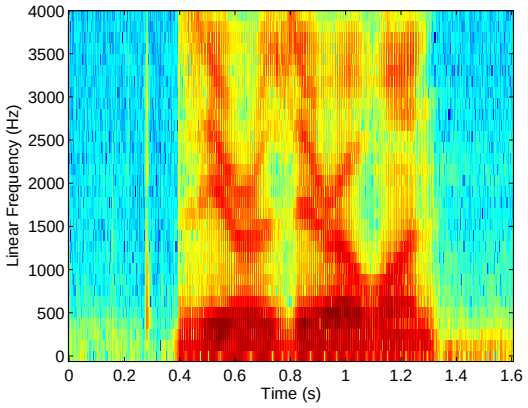
Figure 48. (a) C^4 second-order section (C^4 SOS). (b) Contribution of each individual C^4 in a C^4 SOS and the output of the overall C^4 SOS. (c) An array of 32 C^4 SOSs programmed to have exponential spacing in the center frequencies. The "spikes" in the stopband at approximately 25Hz and 100Hz were due to a systemmatic measurement error using a Lock-In Amplifier. These "spikes" do not represent problems with the circuit, but they illustrate the difficulty in measuring circuit performance as a function of very low frequencies. Longer settling and measurement time constants would improve the accuracy of this measurement, but these low-frequency measurements will still be hard to attain.

very important. Figure 48b shows the contributions of the two stages and how they combine to create the response of the overall C^4 SOS.

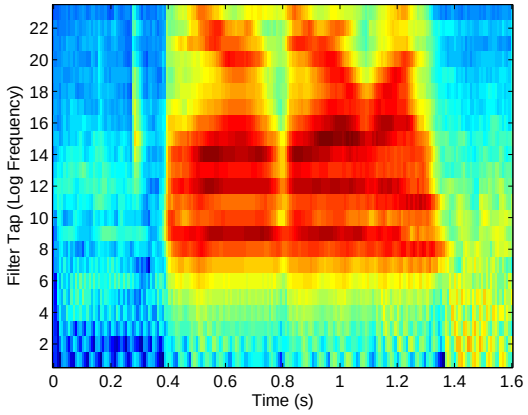
By adding a second row of C^4s to the programmable filter bank, as was shown in Figure 4, an array of C^4 SOSs is formed. When programming this bank of C^4 SOSs, making sure that the time constants of the first and second stage are closely matched is extremely important. If the time constants are not closely matched, then the center frequency spacing will not be even, the bandwidths would greately vary, and the gains could be wildly different from one filter tap to the next. Therefore, programming using FG transistors is essential to attaining a useable filter bank of higher order filters. Figure 48c shows that this higher-order filter bank can, indeed, be programmed accurately. Hence, the programmable filter bank is able to attain very accurate precision in setting the time constants, moderate amounts of resonance for better isolation of the center frequencies, and second-order slopes to improve the rejection in the stopband. In addition, the sharper slopes more closely resemble biology in that the high-frequency slopes are sharper, and the low-frequency slopes now account for the differentiation of the hair cells.

6.4 Programmable Filter Bank for Low-Power Frequency Decomposition

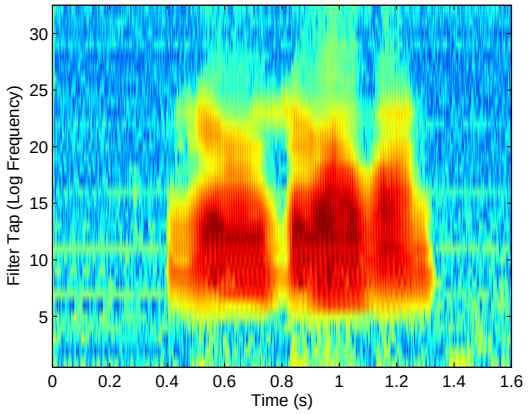
The programmable filter bank provides a high-quality frequency decomposition with a minimal amount of power consumption. To give an example of the PFB's performance, a speech waveform from the TIDIGITS database was used as an input to the PFB that was programmed to have logarithmically spaced center frequencies; therefore, the PFB performed a log-based frequency decomposition of the speech waveform, as is shown in Figure 49c. Very little speech information is contained in the very low (below 70Hz) and very high (above 4kHz) frequencies, as is consistent with common understanding of speech signals. Additionally, the TIDIGITS database



(a)



(b)



(c)

Figure 49. Spectogram response of the filter bank to a speech waveform. The speech signal consists of a male voice saying, "Zero, zero, one" from the TIDIGITS database. (a) Linearly spaced frequency decomposition using a digital Fourier transform (FFT). The maximum frequency is 4kHz, which is half the sampling frequency of the speech signal. (b) Ideal logarithmically spaced frequency decomposition using a bank of ideal bandpass filters. The maximum frequency is 4kHz, which is half the sampling frequency of the speech signal. (c) Logarithmically spaced frequency decomposition using the programmable filter bank. The output of the PFB is very close to the ideal logarithmically spaced frequency decomposition.

has a sampling frequency of only 8kHz, so no information is contained at the high frequencies.

Figure 49a-b are provided for comparison of the output from the low-power hardware with the ideal case (using software). In Figure 49a, the frequency decomposition is performed digitally using an FFT, which is a linearly spaced frequency decomposition. As a result, the speech signal appears to cover a wider spectrum. However, this is an artifact due to the linear spacing; the FFT can be loosely viewed as a filter bank with linearly spaced center frequencies and uniform bandwidths (as opposed to constant Qs). As a result, the lowest frequency "filter tap" covers a very large relative frequency range, and it therefore reaches up to the lower edge of the actual speech content. Also, the highest frequency band extends only to 4kHz since that is half the sampling frequency. As a result, the frequency content appears to be more spread than the log-based approach of either Figure 49b-c. Figure 49b shows the result of an idealized bank of logarithmically spaced, constant Q, bandpass filters. This idealized case is very close to the actual output of the hardware PFB. Again, since the sampling frequency is 8kHz, the output of this ideal filter bank only extends to 4kHz.

As can be seen from the spectograms of Figure 49, the PFB performs a log-based frequency decomposition on par with the idealized case. However, the analog frequency decomposition uses only a fraction of the power required in digital systems. Computing an estimate of the power savings associated with performing a frequency decomposition in analog can be computed as follows. Using the analytic expression for the power consumed by each individual bandpass filter as given by (31), which was presented in Chapter 3, an entire filter bank with 32 subbands consumes $\leq 5\mu$W.

To achieve a similar level of performance digitally using an FFT with 32 subbands and operating at 44.1kHz requires approximately 50MMACS. Using some of the most power efficient DSPs on the market that operate at $4 - 10$MMACS/mW [4], similar computations require approximately 5mW. The analog block, therefore, yields a

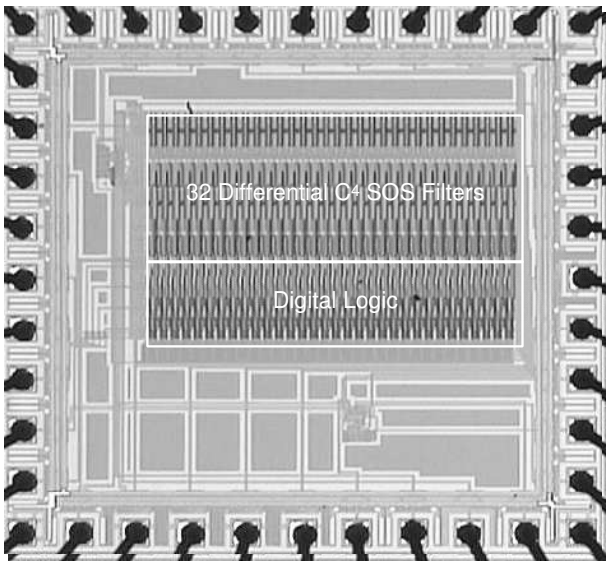


Figure 50. Die photgraph of the programmable filter bank. The dimensions of this tiny chip are 1.5mm ×1.5mm. The die area consumed by the programmable filter bank is comparable to the die area consumed by the resistively tuned filter bank. The larger die area for the programmable filters is because they are differential and, therefore, double the size of the single-ended versions of the resistively tuned filter bank. Also, the capacitor sizes are much larger for the programmable filter bank so that the C^4s more closely match the characteristics of the human cochlea. Only half of the digital circuitry that is highlighted is needed in this design; the functionality of the second half of the digital circuitry was deemed unnecessary after fabrication and could easily be removed for a more compact design.

power savings on the order of a factor of 1000.

Because of the very low-power nature of this analog bandpass array, a spectrum of opportunies are available, including opportunities in high-quality hearing aids which aim to operate with no more than 1mW of power. Using this analog filter bank allows tremendously more signal processing to be conducted, even in the digital domain, so that complex algorithms may be implemented while still meeting the power budget of a hearing aid.

In addition to consuming very little power, this programmable filter bank consumes very little real estate, as well. Figure 50 shows a die photograph of the programmable filter bank. Comparing this filter bank to the resistively tuned version, whose die photgraph was shown in Figure 29, the programmable version consumes a

comparable amount of die area. Since the programmable versions were differential, twice the number of single-ended C^4s were used in the programmable version. Additionally, only half of the digital control circuitry that is highlighted in Figure 50 was required for this design; therefore, this design could be even more compact if needed.

CHAPTER 7

SYSTEM APPLICATIONS

The twofold purpose of this research project has been to develop analog circuitry that behaves in a similar fashion to that of the human cochlea, and in the process, to create a useful, low-power audio front end. Both the biological inspiration and also the pure signal processing permit a wide variety of applications and uses.

The hope for the rest of this chapter is to elucidate some of the capabilities of this programmable analog audio front end. The focus of this discussion will be on several of the applications that are already utilizing the design of this programmable filter bank (PFB). However, this PFB can be used as a springboard for creating a wide variety of other applications besides the ones that will be discussed. To ease the design of these newer applications, the PFB has been placed into a reconfigurable architecture, as will be shown. As a result, we have developed not only a useful audio prototyping platform, but we also have a functional reconfigurable audio system that is able to perform many audio processing algorithms on a single IC.

7.1 Cochlear Modeling

While the programmable filter bank itself is a valid model of the human cochlea since it allows bandpass filters to have logarithmically spaced center frequencies and large amounts of resonance, further attributes of the cochlea can be modeled to provide a more complete model of the human cochlea. Specifically, two attributes of the operation of the cochlea that will yield improved signal processing capabilities are the lateral coupling between neighboring portions of the basilar membrane that help to sharpen the response of the basilar membrane [11] and the adaptation of the cochlea in response to sounds of differing intensities [66]. The following includes a discussion of how we have constructed systems to model these aspects of the cochlea.

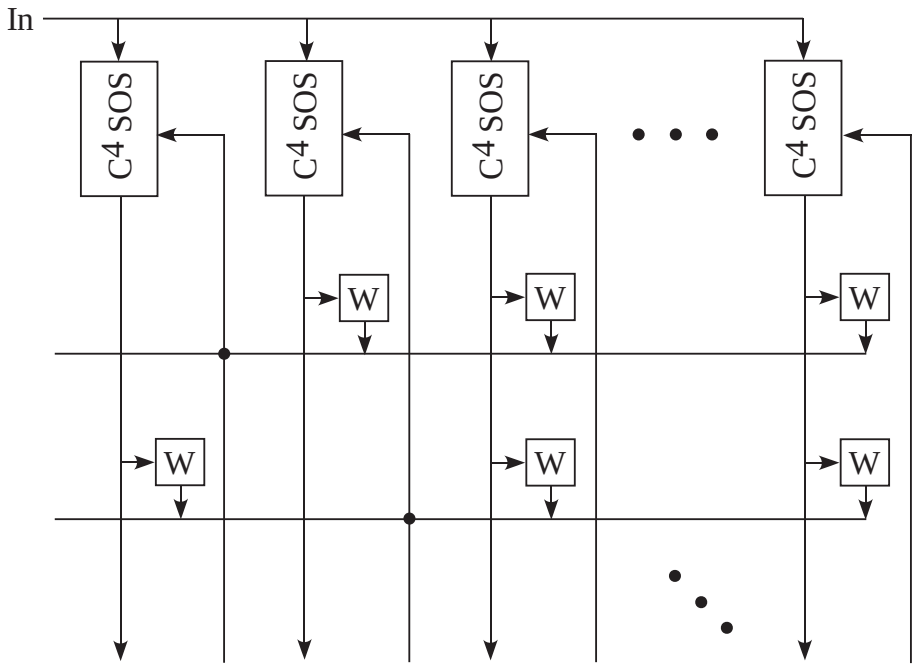


Figure 51. Lateral coupling to model fluid coupling within the cochlea. A weighted sum of the output of the neighboring filters are subtracted from the input of the filter of interest to sharpen the frequency response of the filter of interest.

7.1.1 Lateral Coupling

When the basilar membrane undergoes motion in response to an incident sound wave, the motion of the basilar membrane causes a flow in the fluid surrounding it. This fluid flow partially inhibits the response of neighboring portions of the basilar membrane, therefore sharpening the response of the basilar membrane [11].

The electrical equivalent of this lateral coupling is depicted in Figure 51. In this system, a weighted sum of the outputs of each neighboring stage are added to the input of each individual filter. The result is an effective subtraction of the response of the neighboring filters and, thus, a greatly sharpened frequency response from the filter of interest.

While a weigthed subtraction on both the low- and high-frequency sides of each particular center frequency would help to sharpen the transistions to the stopband, this lateral coupling is most prominent in the frequencies above the center frequency

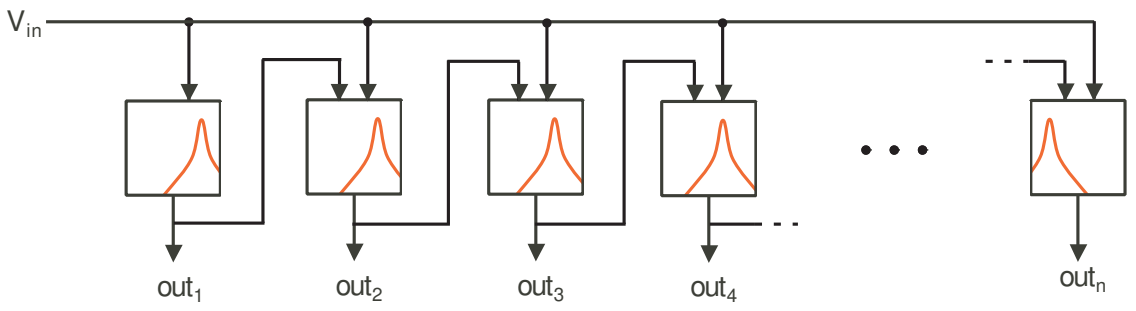


Figure 52. Lateral coupling of higher-frequency filters becomes a cascade of filters. Large cascades lead to accumulation problems, such as noise, and should, thus, be avoided. However, this cascade approach should not have as severe problems with accumulation since the lateral coupling has a relatively small gain as compared to the input each filter receives from V_{in}. Also, only a small range of frequencies are passed to the neighboring stages, and, thus, only the noise from those frequency bands are passed to the neighboring stages. Additional lateral coupling can be used from additional stages preceeding the filter of interest. As a result, the filter may receive lateral coupling from several of the previous filters.

in biological systems [11]. If only the higher frequencies are to be coupled in, another way to view this system would be what is shown in Figure 52. Each filter in the array receives the common input to the entire system and also the output from the previous stage, which has a slightly higher center frequency. Added input lines could also come from previous stages, as well; however, too many added input lines could cause difficulties in designing the IC for fabrication.

The major problem with this type of approach is clearly evident from Figure 52. One major advantage of the array of bandpass filters in parallel was the minimization of accumulation problems that came as a result of a cascade. In the early cascade models [20, 11, 14, 13], significant problems resulted from cascading many lowpass filters, as was described in Chapter 2. The design of Figure 52 is basically a cascade that also has a common input. This design will also suffer from similar cascade problems, even though it will not be as severe since the bandpass filters are passing only the frequencies and noise within their own particular passband.

This lateral coupling can, however, be achieved with a minimal amount of cascading. This "lateral coupling" can be done by adding an extra "dummy" filter along

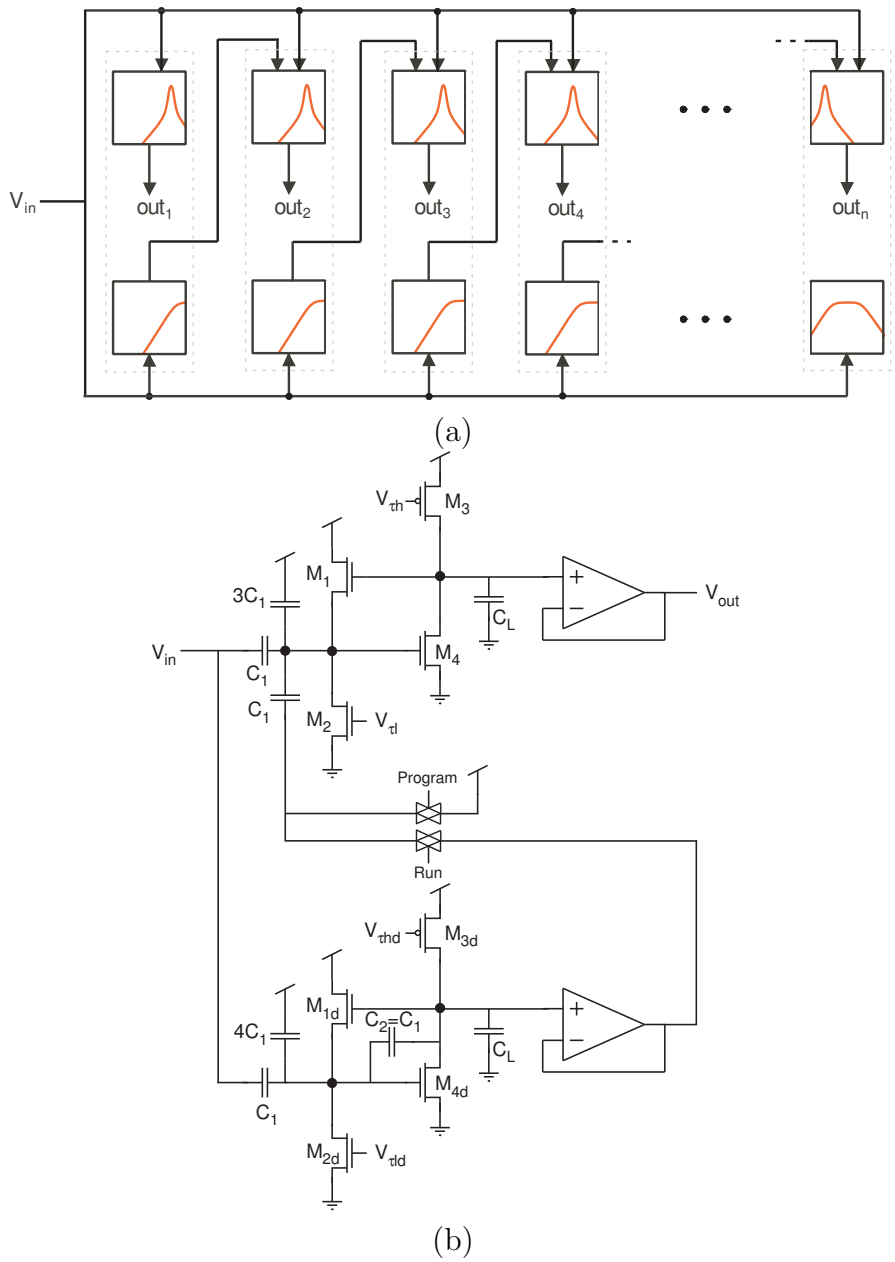


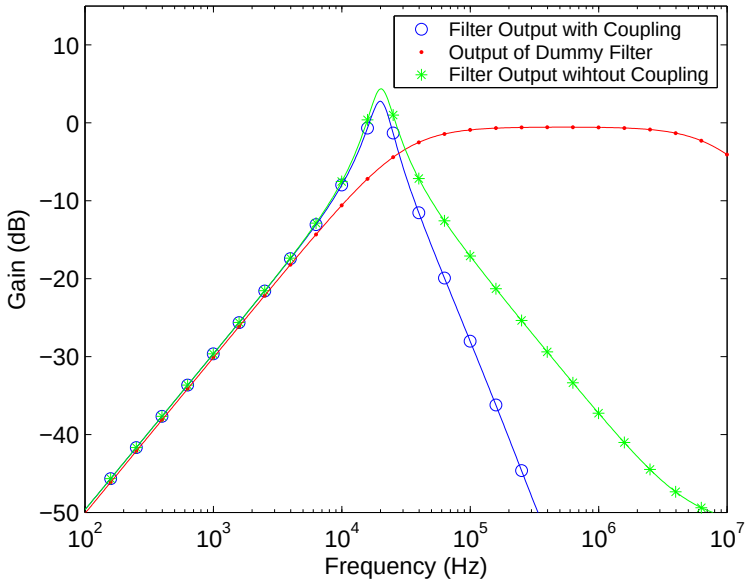
Figure 53. Lateral coupling of higher-frequency filters with "dummy" filters. (a) Each filter tap includes an extra filter that is tuned to have a wider bandwidth. This dummy filter receives the same input, V_{in}, as the other filters, but only the dummy filter passes its output to the next filter tap. As a result, this technique has no cascade longer than two filters at any given point. Also, the wide-band filter produces the effect of several cascaded filters since it represents the sum of the preceeding filter stages. The dashed boxes represent each individual filter tap with its associated dummy filter. (b) Schematic of the effective circuit for each filter tap. This schematic shows the C^4 of the representative filter tap and the dummy filter from the next higher-frequency stage. The main C^4 receives a capacitively coupled input from the dummy filter. The second input capacitor is connected to V_{dd} during programming, or tuning, mode so that it effectively adds to the total value of C_W, and the corner frequency will not change from tuning mode to run mode.

with each filter tap that receives the same input as its associated filter but sends its output to be subtracted from the next stage, as is shown in Figure 53a. This dummy filter is a lower-Q filter than its associated bandpass filter. Also, this filter is set to have a wider bandwidth than its associated filter so that it approximates several of the neighboring filters since the sum of all the higher-frequency filters is essentially the same as a single wide-band filter. The circuit-level schematic of this type of configuration is shown in Figure 53b. Specifically, this schematic represents the C^4 from the filter tap of interest and the wide-band, lower-Q C^4 from the next higher-frequency stage. The buffered output of the dummy filter is capacitively coupled to the main C^4. During a tuning, or programming, phase, this added capacitor is connected to V_{dd} so that it effectively adds to the capacitance of C_W; therefore, when the run-time phase is turned on, the effective capacitance seen by each time constant remains fixed, and the corner frequencies do not shift from programming mode to run mode.

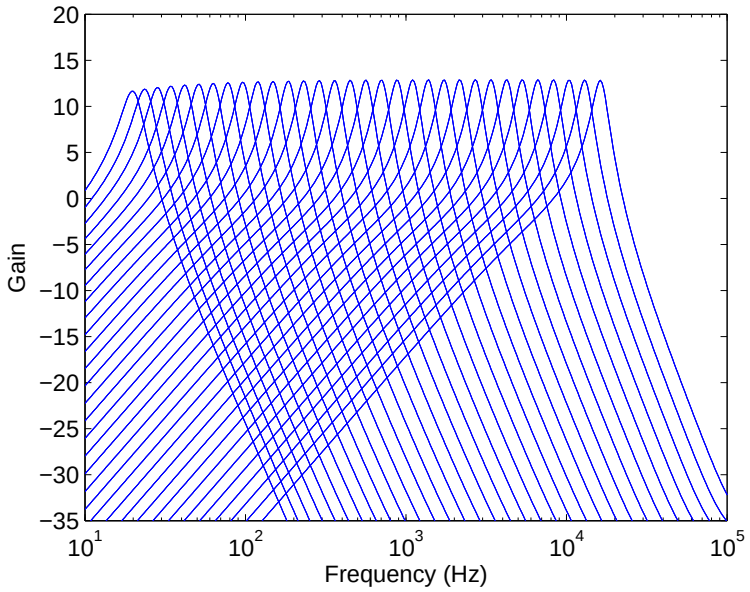
Figure 54 shows the simulated results of the circuit of Figure 53. Both the output of a single stage of this type is shown, as well as the output of an entire array in this confugration. Included in the frequency response of the single stage (Figure 54a) is also the frequency response of a C^4 under the same biasing conditions that does not receive lateral coupling. As can be seen, the laterally coupled C^4 has a much faster high-frequency roll off than does the nominal C^4 (-40dB/decade as opposed to -20dB/decade). Also, the output of the dummy filter is clearly shown to be a wide bandwidth response, which simulates the effect of laterally coupling several of the higher-frequency stages. This circuit is currently being fabricated in a 0.5μm process available through MOSIS.

7.1.2 Q-Adaptive Systems

In order to both protect the ear from the hazards of loud sounds and also to increase the dynamic range through which a person can hear, the human ear has several



(a)



(b)

Figure 54. Response of the C^4 to lateral coupling. Using the mechanism shown in Figure 53, the high-frequency roll off for a C^4 goes from -20dB/decade to -40dB/decade. (a) The output of a single filter tap. Shown here are the output of the C^4 and also the "dummy" filter that is coupling into the C^4. For comparison, a C^4 without any lateral coupling is shown, and this comparison clearly shows the increased high-frequency roll off due to lateral coupling. (b) An entire array of laterally coupled C^4s. As is clearly evident from this plot, the high-frequency slopes are much steeper than the low-frequency slopes. The high-frequency slopes are now -40dB/decade, instead of -20dB/decade.

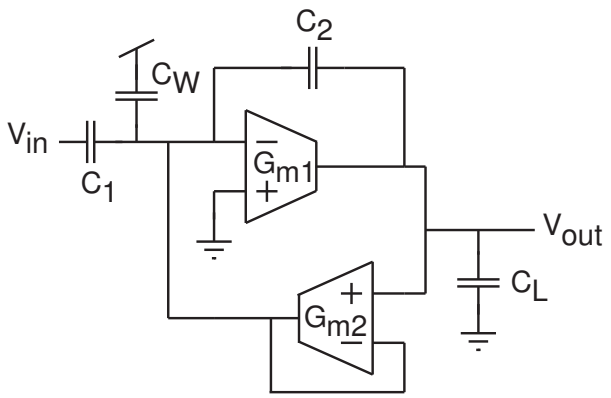


Figure 55. Generalized schematic of the C^4. Each transconductance element (G_{m1} and G_{m2}) can be replaced with the appropriate transconductance element. The basic C^4 replaces G_{m1} with a common-source amplifier and G_{m2} with a source follower.

mechanisms to alter the sensitivity of its response [66]. In particular, the outer hair cells lengthen or contract in response to soft or loud sounds, respectively, to maximize the effectiveness of the sound transduction by the inner hair cells [67]. In a process known as outer hair cell motility, the outer hair cells determine the amount of movement the inner hair cells can undergo and, thus, the amount of stimulation the inner hair cells receive. Viewed in another way, outer hair cell motility alters the effective resonance, Q, of the bandpass filter [17].

In order to increase the dynamic range of the bandpass filter, or, similarly, vary its Q with varying input sound levels, a wide variety of options exist. Typically, this type of system is considered automatic-gain control (AGC) and can thus receive feedback from other parts of the system to ensure that the input signal level always remains within a close proximity of the optimal input-signal level [14]. While this type of system can often perform adequately, it fails to capture the dynamical properties within the cochlea itself, and it can lead to a significant increase in overhead. However, by placing as much of the computational power within the filter itself, the filter can take advantage of its dynamical properties to perform the necessary functions.

The C^4, which has been the basic bandpass filter for all aspects of our cochlear model, can be redrawn as shown in Figure 55. This "generic" version of the C^4 uses

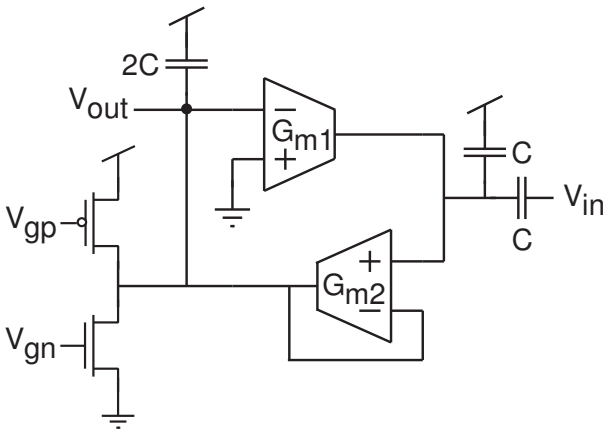


Figure 56. Bandpass filter based upon the C^4 circuit topology that uses its inherent nonlinearites for Q adaptation. The push-pull stage helps to achieve sinh **properties.**

a general high-gain inverting amplifier in place of the common-source amplifier in the C^4, and it also uses a general unity-gain buffer from the output back to the middle node in place of a source-follower. These transconductance elements can be replaced with a wide variety of elements as long as they meet the requirements of high-gain inverting amplification or unity-gain amplification. As a result, the C^4 can be viewed as more than simply a single version of a bandpass filter, but it can now be viewed as an entire family of bandpass filters.

By replacing the transconductance elements with ones that provide a given amount of dynamics, such as sinh properties, the amount of resonance within the filter can be controlled by the transconductance elements, themselves. For example, the circuit of Figure 56 is one such version of the C^4 in which the dynamical properties of the C^4 can help to achieve Q adaptation. The push-pull stage helps to achieve sinh properties.

7.2 Applications of the Programmable Filter Bank

A wide variety of signal-processing applications can be performed by using the PFB as a front end or as a smart sensor interface. In the following discussion, the focus will be placed upon noise-suppression and speech-recognition systems, which are both interesting problems that are yet to be solved in marketplace technologies and are also

two of the applications that we have concentrated upon. Additionally, the low-power PFB can (and will) be used in developing embedded sensors, such as hearing aids and cochlear implants.

7.2.1 Noise Suppression for Speech Enhancement

The ability to remove additive background noise from an audio signal has recently received increased attention due to the prosperity of portable communication devices. By using the PFB as an initial processing step, we have designed a real-time, low-power technique for noise suppression in the continuous-time domain, as shown in Figure 57. The goal is to design a real-time system that generates an optimal estimate of the actual signal from an additive mixture of signal and noise. We assume that the additive noise is constant over a long time period relative to the transient patterns of speech. The noisy signal is separated into 32 bands that are exponentially spaced in frequency, as can be achieved by the PFB. Then, a gain factor is calculated based on the envelopes of each observed subband speech signal and subband noise signal, which serves to estimate the SNR of the incoming signal in that band. Bands with low SNR are attenuated while bands with high SNR pass through unatenuated. The band-limited signal is multiplied by the gain factor, and the result of all 32 bands are summed together to reconstruct the full speech waveform with the additive noise components suppressed.

The gain calculation block, as shown in Figure 57b, first estimates the levels of both the noisy signal and the noise. The noisy-signal envelope is estimated using a peak-detector circuit, and the noise level is estimated using a minimum detector operating on the noisy-signal envelope at a slower rate. Currents that represent the noisy-signal level and the noise level are divided using a translinear division circuit to create an output current that is an estimate of the SNR. An optimal weiner gain function is used in combination with the estimated SNR to calculate each gain factor. Further circuit details and signal-processing theory relating to this algorithm have

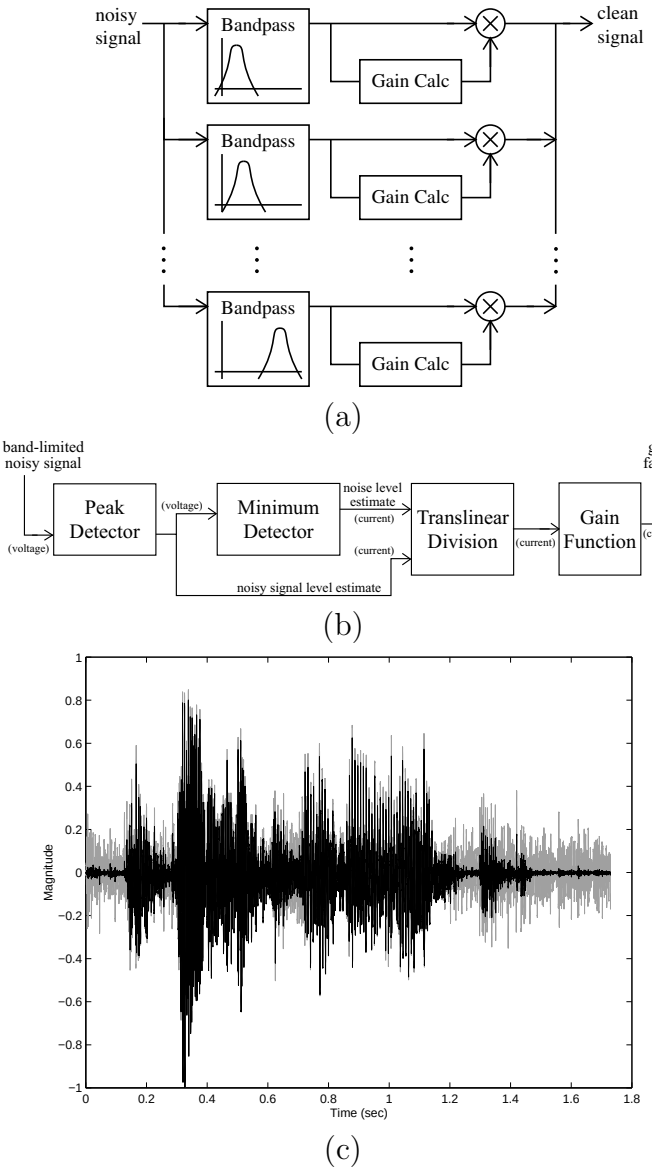


Figure 57. Continuous-time noise-suppression system. (a) Overall structure of the system. The incoming noisy signal is divided into exponentially spaced frequency bands using the PFB. Next, the optimal gain for each band is computed. If the band has sufficient estimated SNR, then the signal passes through with maximum gain; otherwise, the gain is reduced dependent upon the the estimated SNR in that particular band. The resulting gain factor is multiplied with the band-limited noisy signal to produce a band-limited "clean" signal. Finally, the output of all of the bands are summed to reconstruct the signal with the noise components significantly reduced. (b) Gain calculation block. Within each frequency band, the noisy signal envelope is estimated using a peak detector. Based upon the voltage output of the peak detector, the noise level is estimated using a minimum detector operating at a slower rate than the peak detector. The currents representing the noisy signal and noise levels are input to a translinear division circuit, which outputs a current representing the estimated SNR. A nonlinear function is applied to the SNR current to calculate a gain factor. (c) Experimental measurements of noise suppression. The light gray data is the subband noisy-speech input signal; the black waveform is the corresponding subband output, after the gain function has been applied.

been presented in detail elsewhere [8, 68, 69].

Figure 57c shows a noisy speech signal that has been processed by this system. As can be seen, this system is effective at adaptively reducing the amplitude of noise-only portions of the signal while leaving the desired portions relatively intact. Additionally, any noise or distortion created by the gain calculation circuits minimally affects the output signal because these circuits are not directly in the signal path. While the bandpass filters and the multipliers will inject a certain amount of noise into each frequency band, this noise will be averaged out by the summation of the signals at the output of the system.

7.2.2 Speech Recognition

The PFB can be used as the front end to a wide variety of speech recognition systems. The following is a brief overview of the two major speech recognition systems that have been designed in conjunction with the PFB.

7.2.2.1 Continuous-Time Cepstrum Encoding

The mel-cepstrum is often computed as the first stage of a speech recognition system [70]. Figure 58 shows a block diagram for an analog cepstrum, which is an approximation to either the mel-cepstrum or cepstrum. The speech waveform is initially decomposed in a logarithmically spaced fashion using the PFB. As a result, the output of the filter bank contains information similar to a Fourier transform and, therefore, respresents the product of the excitation and vocal-tract within that filter band. The primary difference between the digital and analog versions is that the digital mel-cepstrum approximates the logarithmic frequency content of the human ear by combining discrete-Fourier transform bands while the analog version performs an actual band-like analysis of the input signal. Thus, higher-frequency critical-band energies are effectively computed using shorter basis functions than the lower-frequency

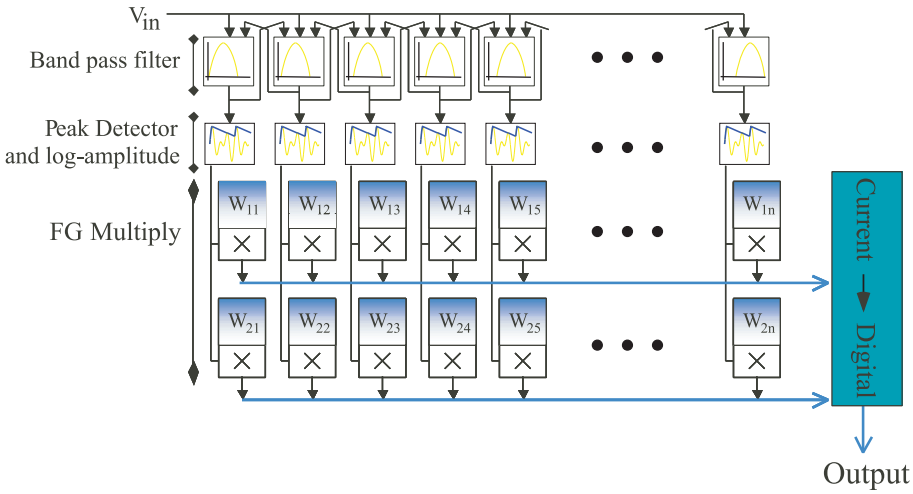


Figure 58. Speech recognition using cepstrum encoding. A logarithmically spaced frequency decomposition is performed with the PFB followed by a peak detection. The discrete cosine transform is performed by using an array of floating-gate circuits.

bands. Consequently, the analog version is closer to the operation of the human auditory system and is better suited to identifying transients. A detailed discussion of the signal-processing foundation of analog and digital mel-cepstrum computations has been presented elsewhere [9, 71]; the primary difference between the analog- and digital-computation approaches is in the frequency decomposition and amplitude dection method.

The continuous-time cepstrum begins with a frequency decomposition using the PFB. The magnitude function is approximated using a peak detector rather than using the true magnitude of the complex spectrum. The discrete cosine transform (DCT) is performed by using a matrix multiply utilizing an array of floating-gate circuits in which each row of the matrix is a DCT basis vector.

7.2.2.2 Biologically Inspired Feature Extraction

Additional speech-recognition systems can be built that are based upon biological models of the human auditory system. One such biological model [72, 73] consists of three stages. The first stage models the filtering done by the cochlea and, therefore, performs a logarithmically spaced frequency decomposition, as can be done by the

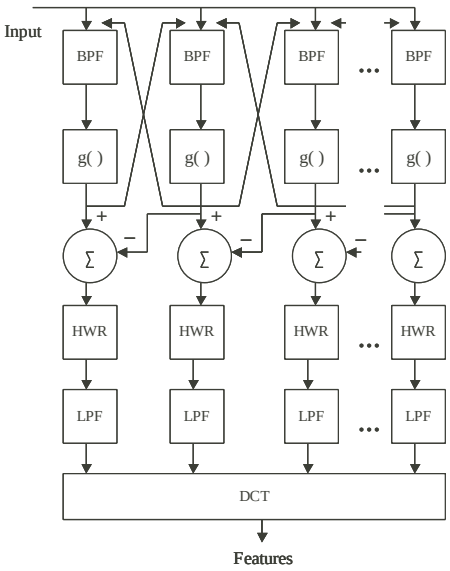


Figure 59. Biologically inspired feature extraction for speech recognition. The PFB initially decomposes the speech waveform into individual subbands and is used to model the filtering properties of the cochlea. This feature-extraction system also models the transduction of the inner hair cells and the lateral inhibition of the cochlear nucleus. Further details of this speech recognition model can be found elsewhere [3].

PFB. The second stage models the transduction process of the inner hair cells and, therefore, performs a time derivative, a non-linear compression, and a lowpass filter. The third stage models the lateral inhibitory network in the cochlear nucleus and, therefore, consists of a spatial derivative, a spatial low-pass filter, a half-wave rectifier and a temporal integrator. Figure 59 illustrates this biologically inspired feature-extraction model. Further details of this speech-recognition system, including details of the analog circuitry used in building this, can be found elsewhere [3].

7.3 Reconfigurable Analog Architectures

While designing and building analog systems for very-low power and high-quality systems is beneficial for portable electronics and embedded sensors, the design process involved with any of these analog systems is very long and requires a certain amount of expertise. One reason why digital systems are so prevalent is that they are easy to use, and they can often be prototyped on reconfigurable architectures such as

field programmable gate arrays (FPGAs). If analog systems are to be a truly viable alternative to digital systems in marketplace technologies, then the design and use of analog systems must be made intuitive and friendly.

Additionally, the time-to-market involved with analog circuitry must be decreased if analog systems are to compete with digital systems, since digital systems can easily use FPGAs to synthesize and test hardware implementations. Typically, analog systems must undergo a lengthy design and simulation phase accompanied by a long (approximately 3 month) fabrication phase. If the first silicon implementation does not fully meet the given specifications, then the system must be redesigned and refabricated, as is illustrated in Figure 60. In addition to being a lengthy process, the multiple design and fabrication iterations are extremely costly.

To increase the ease of analog-circuitry design and to also decrease time-to-market associated with an analog design, a large focus has been placed on building reconfigurable analog architectures [74, 58, 75]. Analogous to field programmable gate arrays, these reconfigurable analog architectures, called field programmable analog arrays (FPAAs), are being developed to speed up and also ease the burden of analog-circuit design. Instead of the fabrication stage consuming the vast majority of the time in the analog-design cycle, the focus can be placed on working with actual hardware implementations of algorithms by synthesizing the algorithm into analog hardware, as is shown in Figure 60.

Figure 61 shows the general architecture for our FPAAs. Each of the analog circuit elements that are to be used as the basic building blocks for the hardware synthesis are placed into configurable analog blocks (CABs), and the CABs are, in turn, placed into a large array. Within each CAB is a wide variety of analog-circuit blocks ranging in granularity. The actual circuit elements include basic elements (capacitors and transistors), medium-sized elements (OTAs), and more-complex elements (filters and vector matrix multipliers) [74]. As a result, the FPAA is very general and allows for

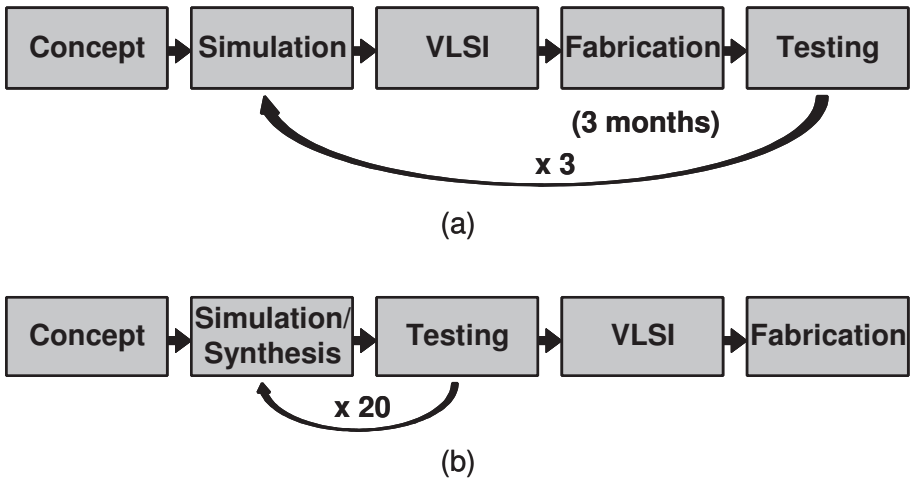


Figure 60. Design cycle using standard analog-design methods and using FPAAs. (a) Design cycle of an IC using standard analog-design methods. Designs require a large amount of simulation and, typically, several iterations of fabrication, which is expensive and a lengthy process. (b) Design cycle of an IC using an FPAA. Most of the design time involves simulation and testing in hardware, which can be done very quickly. If needed, only a single fabrication step is performed.

a wide variety of algorithms to be synthesized on it.

Floating-gate transistors are used as the basic switching elements connecting individual analog elements together within a CAB and also connected neighboring CABs together. Additionally, floating-gate transistors are used to bias individual elements within the CABs, including OTAs and C^4s. Beyond simply the circuit elements within the CABs, the actual switch matrix, which is composed of an array of floating-gate transistors, can be used to synthesize floating-gate circuits such as voltage references.

While many systems can be developed using a general FPAA, the addition of switches between each individual element can seriously hamper the performance of a system. In fact, it is this reason that the general FPAA uses a mixed granularity of elements (from transistors and capacitors to tunable filters and multipliers) instead of simply using only transistors and capacitors.

However, this granularity can be expanded out to include even more useful functional blocks. If certain blocks are used in virtually all algorithms of a certain type, then that block could also be included as its own entity, even if it could be compiled

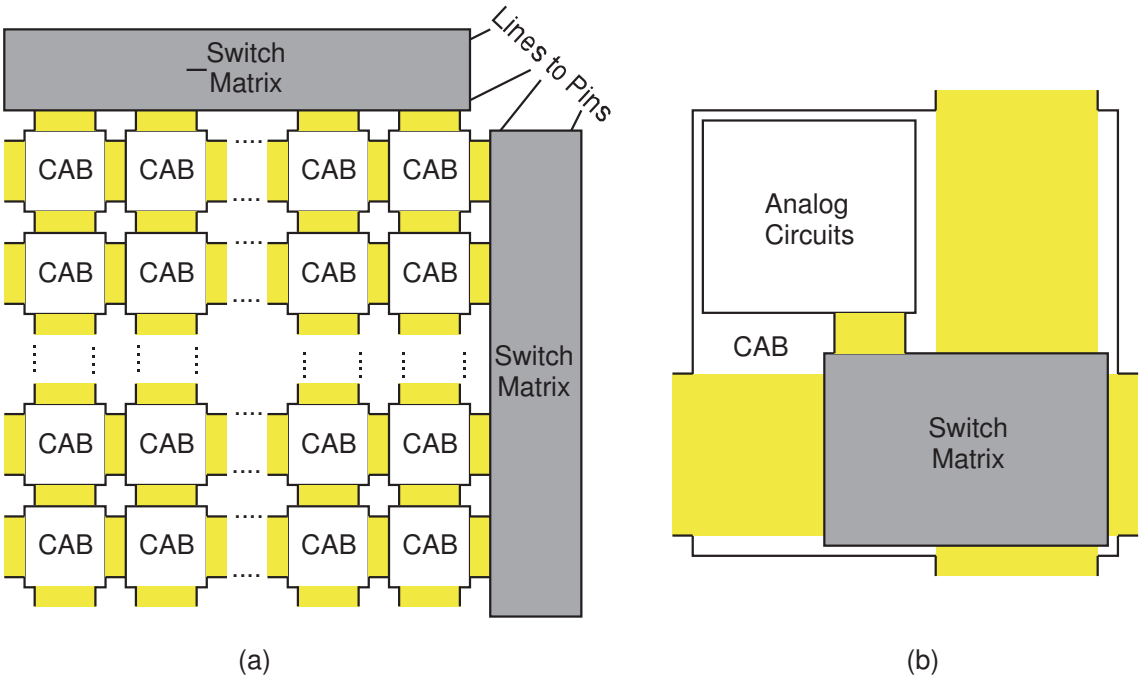


Figure 61. Block diagram of the general FPAA chip. (a) Architecture of the general field programmable analog array. The array consists of an array of configurable analog blocks (CABs) which each contain a wide variety of basic analog building blocks. In the general FPAA IC, the CABs contain analog devices of mixed granularity, including small elements (capacitors and transistors), medium-sized elements (OTAs), and more complex elements (filters and vector matrix multipliers). (b) Signal flow in a single CAB. Each of the analog-circuit elements in the CAB are connected together via an array of floating-gate switches. These floating-gate switches also connect one CAB to its neighboring CABs.

from the elements within the FPAA. For example, the frequency decomposition is a reoccurring theme in audio-processing algorithms; while it could easily be compiled from the individual CABs of the general FPAA, it would effectively consume a large portion of the resources (interconnects, bus lines, and computation components) of the FPAA and would operate at a quality below its potential due to the added parasitics. However, since this frequency decomposition is so commonly used, a single frequency-decomposition block could be placed on the IC in order to add this functionality, while still enabling other circuits to be compiled.

While this frequency-decomposition block is a useful example in the audio domain, it may not be necessary in another application domain. Essentially, once an FPAA goes beyond a certain granularity in the components that it offers, the FPAA actually becomes application specific. So, if a PFB is to be added to the FPAA, then other high-level elements should be added to the CABs since this design has become application specific to the audio domain. As a result, the individual CABs should include circuits that are often used in audio-processing applications, such as envelope detectors, multipliers, non-linear transform elements, linear combiners, and vector-matrix multipliers, as well as basic components (transistors, capacitors, and amplifiers). While this new FPAA may not be able to perform any general task, it will, however, be able to perform virtually any audio task, and it will be able to do so with increased performance over a generalized FPAA.

As a result, we have designed and fabricated an audio-specific FPAA and have named it the reconfigurable analog signal processor (RASP) 3.0. The 3.0 designation indicates that this chip is essentially the third wave of analog reconfigurable architectures. Specifically, this RASP 3.0 presents a departure from having FPAAs perform general duties and instead looks to them to perform high-quality specific tasks. In essence, this third generation of FPAAs is sacrificing generality to achieve increased performance. In addition to the RASP 3.0, several other application-specific devices

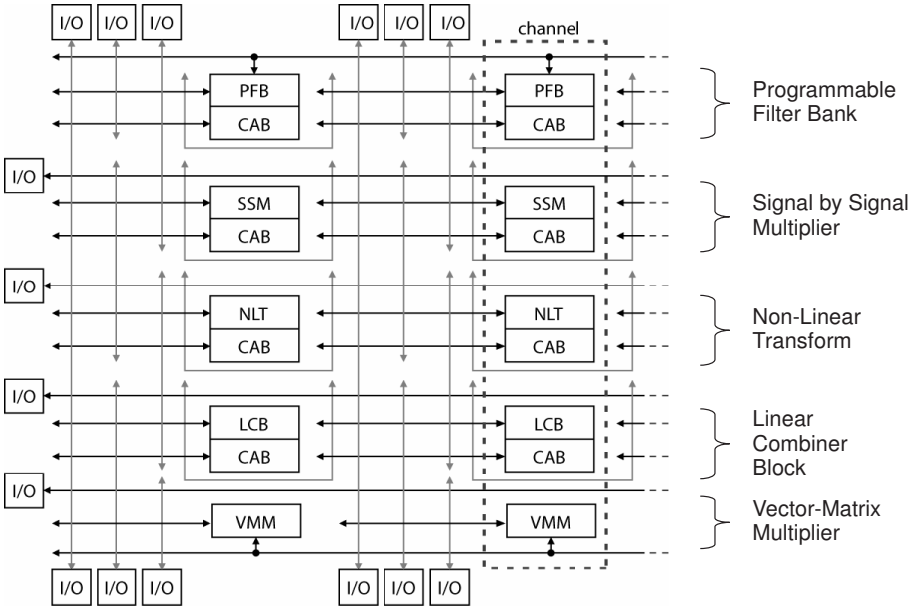


Figure 62. Architecture of the RASP 3.0. This FPAA was designed to be an application-specific reconfigurable system in which the application is audio-signal processing. This chip will be able to perform a wide variety of algorithms including hearing-aid algorithms, noise suppression, and speech-recognition front ends.

are currently being developed including one for neural modeling [76] and one for performing different mathematical and chaotic systems [77].

Figure 62 shows a block diagram of the RASP 3.0, which is a reconfigurable system designed specifically for developing audio-processing systems. Since frequency decomposition is nearly ubiquitous in audio-processing systems, the RASP 3.0 includes the PFB as the first major component. Also, since most audio systems perform the largest portion of the signal processing only on subbanded signals, the overarching design behind this architecture is to perform the vast majority of the signal processing in individual columns. Since certain algorithms call for lateral coupling between neighboring subbands, a certain number of horizontal interconnects are also present. The RASP 3.0 recently returned from fabrication, and Figure 63 shows the die photograph. This large integrated circuit has the dimensions of 4.5mm $\times$ 8mm, and a PCB is currently being designed for it.

Again, since the RASP 3.0 is specifically geared towards audio applications, the

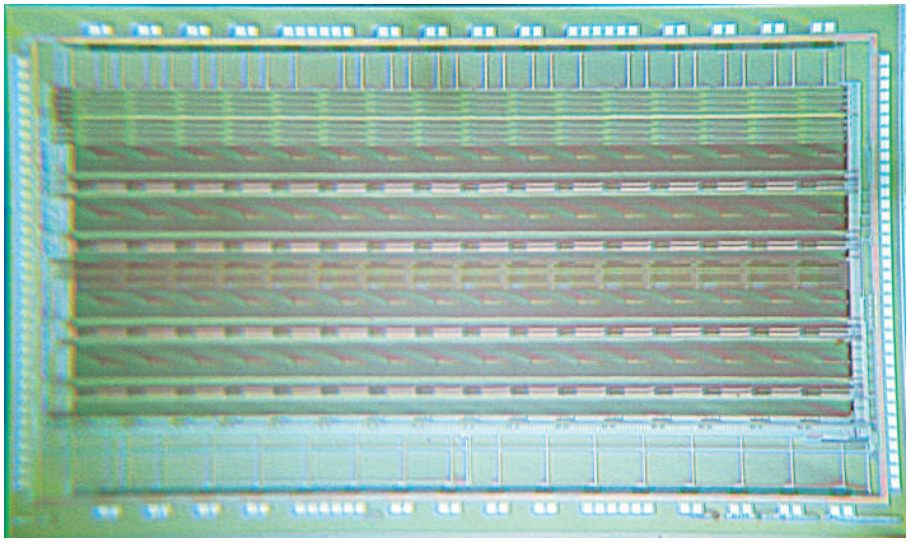


Figure 63. Die photograph of the RASP 3.0. The RASP 3.0 is a reconfigurable analog IC that is specifically geared towards performing high-quality audio-processing tasks. The dimensions of this IC are $4.5\text{mm} \times 8\text{mm}$**.**

specific components that can be switched into and out of a system design should also be circuits that receive regular use in audio processing systems. As can be seen in Figure 62, one of each major functional block is included in each of the columns. These functional blocks include signal-by-signal multipliers, non-linear transform elements, linear combiners, and vector-matrix multipliers. General CABs, which include basic components such as capacitors, transistors, and amplifiers, are interspersed within each column to provide added functionality. Since a new algorithm may be developed that includes a circuit not provided by the functional blocks, the general CAB provides an added degree of flexibility.

The RASP 3.0 will enable rapid prototyping of audio systems as well as provide a useable platform for implementing these same audio systems. While increased performance (reduced parasitics and power consumption) could potentially be achieved by taking the compiled design from the RASP 3.0 and re-fabricating it without all the unnecessary extra switches and components, the high-level of granularity of the RASP 3.0 allows sufficient performance that the RASP 3.0, itself, could be used as

the "final silicon."

Several of the initial applications to be compiled on the RASP 3.0 will be the algorithms already discussed previously in this chapter, including the noise suppression algorithm and the two different speech recognition systems. Additionally, this FPAA will be used to help design better hearing aids and cochlear implants. The correct circuits are in place that will allow several different hearing aid models to be implemented within the RASP 3.0, as well as cochlear implant models.

In addition to incorporating the PFB into the design of the RASP 3.0, indirect programming was also heavily used in this design. Specifically, the switches, which are typically single floating-gate transistors, have been replaced with indirectly programmed floating-gate transistors. Using indirectly programmed FGs has the added benefits of being able to turn on multiple switches at the same time, better isolation of the switches for programming purposes, and the ability to program each switch to a desired level instead of simply on or off. This last benefit is particularly useful when using the switch elements themselves as elements to be used in the system design and not simply just switches. Also, a single FG transistor that shares a FG with two other transistors, can be used to inject (or turn on) both of the switches at the same time. When using fully differential systems, there are switches on both sides of the differential signal that would need to be turned on. By using indirect programming, only one FG must be injected to turn both switches on.

CHAPTER 8
CONCLUSIONS AND FUTURE DIRECTIONS

8.1 Conclusions

In this research, we have been able to use the human cochlea as inspiration to build a better front end for audio processing systems. Specifically, we have taken the resonance nature of the basilar membrane and have used it to inspire an array of programmable bandpass filters that perform a logarithmically spaced frequency decomposition. This frequency decomposition is analogous to a discrete-Fourier transform, but the analog version that has just been presented is able to do the task at roughly one thousandth of the power required in the digital domain.

This cochlear model performs the frequency decomposition in a manner more closely matched to the actual cochlea than did the previous researchers' cochlea models that focused on the traveling wave properties of the cochlea. By using an array of bandpass filters in parallel, this cochlear model more closely agrees with biology than did the cochlear models incorporating a cascade of lowpass filters.

Additionally, due to the use of floating-gate transistors, the accuracy of tuning the bandpass filters to the desired poles and zeros can be set very precisely. We have shown that the filters can be programmed to within a desired tolerance by using a single calibration step. The programming routine presented here can also be applied to virtually any other system utilizing FG transistors for precision analog circuitry.

By simply sharing the FG node of one of these FG transistors with another transistor, we have shown that a large portion of the parasitics associated with precise FG programming can be completely removed. Also, this indirect programming of FG transistors blurs the line between a programming phase and a run-time phase; the two no longer need to be separate, but recalibration can be performed during the run-time phase.

The programmable filter bank has already been shown to be useful in a variety of audio-processing systems; most notably, it has been successfully used in noise-suppression and speech-recognition systems. With the inclusion of the programmable filter bank into a reconfigurable architecture, as we have done with the RASP 3.0, many more applications will be developed in a much shorter time period. This reconfigurable architecture serves as a high-quality rapid prototyping platform because it is composed of real circuits in real silicon and not just a simulation model. Also this reconfigurable architecture can serve as a very useable final product, much in the same way as FPGAs are becoming the final design instead of simply a design platform.

8.2 Future Directions

This programmable filter bank is far from the end of this research project; in fact, it is just the beginning. This programmable filter bank is the key to future research in this field and will hold a prominent place in future low-power, programmable-analog audio designs. This programmable filter bank, especially when combined with a reconfigurable architecture will allow many more designs in far less time.

Using the FPAA as a prototyping tool will allow designs to become more efficient. No longer will designs need several months to simulate, layout, and fabricate, but an algorithm can go from conception to silicon in a matter of hours or days. Being able to utilize the FPAA will greatly aid the design of future audio-processing systems such as sound localization, biometric applications, noise suppression, beamforming, speech recognition, hearing-aid algorithms, and cochlear-implant algorithms.

Also the precision that is available with these floating-gate transistors allows analog to be used in place of digital systems, even when accuracy is required. Additionally, when using MOSFETs in the subthreshold regime, the power consumption is greatly reduced. As a result, analog circuitry can replace digital circuitry, and

very low-power applications can be developed. Consumer electronics are increasingly adding functionality but must also meet the demands of long battery lifetimes. Using low-power, programmable-analog circuitry can provide high functionality while still maintaining low power consumption. Also, embedded sensors typically require long battery lifetimes but must still be able to adequately perform their duties. For example, hearing aids and cochlear implants, which can both make use of the programmable filter bank, must be able to help the user hear better, but cannot draw too much power. Further, a wide variety of implantable medical devices can be imagined in which low power and high quality are competing demands. However, with this programmable analog circuitry, both demands can be met.

While the focus of this research has been in developing a silicon model of the human cochlea, other systems can be built by drawing inspiration from other biological systems. For example, modeling muscle-control processes can yield robotics applications, and modeling the human visual system can improve image recognition processes. Both of these will be future elements of this research.

APPENDIX A

C^4 DERIVATION

A.1 Derivation of the C^4 Equations for Cochlear Modeling Purposes

The following derivations assume that the C^4 is being operated in the fashion typically used in cochlear modeling. Specifically, the corner frequencies are assumed to be close to each other so that there is a narrow passband. Also, the transistors are assumed to be operating in the subthreshold regime. Following this section will be a general derivation of the transfer function of the C^4 that holds for operating in the subthreshold regime, as well as moderate and strong inversion.

A.1.1 Derivation fo the C^4 Transfer Function

The schematic for a C^4 is shown in Figure 67. The derivation of the transfer function of a C^4 begins by finding the differential equations for the two nodes, which are

$$\begin{aligned} C_1\frac{d(V_{in}-V_{fg})}{dt} + C_W\frac{d(-V_{fg})}{dt} + C_2\frac{d(V_{out}-V_{fg})}{dt} &= I_{bl}[1-e^{(\kappa\Delta V_{out}-\Delta V_{fg})/U_T}] \\ C_2\frac{d(V_{fg}-V_{out})}{dt} + C_L\frac{d(-V_{out})}{dt} &= I_{bh}[e^{\kappa\Delta V_{fg}/U_T}e^{\Delta V_{out}/V_A}-1] \end{aligned} \tag{60}$$

However, the simplification that

$$e^{\Delta V_{out}/V_A} \approx 1$$

applies since the change in the output voltage, ΔV_{out}, will be much smaller than the Early Voltage, V_A. Reorganizing the terms, the equations become

$$\begin{aligned} C_1\frac{dV_{in}}{dt} + C_2\frac{dV_{out}}{dt} - (C_1+C_2+C_W)\frac{dV_{fg}}{dt} &= I_{bl}[1-e^{(\kappa\Delta V_{out}-\Delta V_{fg})/U_T}] \\ C_2\frac{dV_{fg}}{dt} - (C_2+C_L)\frac{dV_{out}}{dt} &= I_{bh}[e^{\kappa\Delta V_{fg}/U_T}-1] \end{aligned} \tag{61}$$

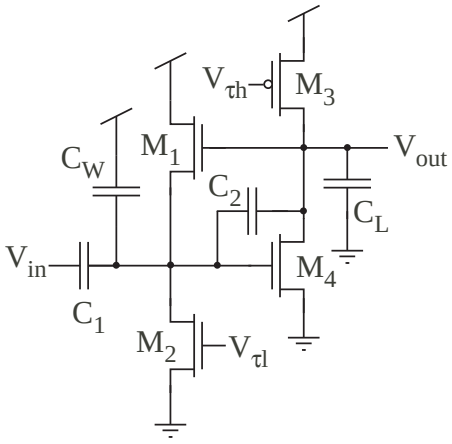


Figure 64. Schematic of the C^4. This is an nFET version of this circuit. A pFET version can also be made by simply flipping the circuit and changing all nFETs to pFETs and all pFETs to nFETs. The overall transfer function is identical for both cases. Only the signs of voltages vary in the initial differential equations.

Further simplifications can be made to the right-hand side of both of the node equations by using the following approximations:

$$e^x - 1 \approx x$$

$$1 - e^{-x} \approx x$$

This transforms (61) into

$$\begin{aligned} C_1\tfrac{dV_{in}}{dt} + C_2\tfrac{dV_{out}}{dt} - (C_1 + C_2 + C_W)\tfrac{dV_{fg}}{dt} &= -I_{bl}\tfrac{\kappa\Delta V_{out} - \Delta V_{fg}}{U_T} \\ C_2\tfrac{dV_{fg}}{dt} - (C_2 + C_L)\tfrac{dV_{out}}{dt} &= I_{bh}\tfrac{\kappa\Delta V_{fg}}{U_T} \end{aligned} \tag{62}$$

By taking the Laplace transforms of (62), these equations become

$$\begin{aligned} sC_1V_{in} + sC_2V_{out} - s(C_1 + C_2 + C_W)V_{fg} &= -\tfrac{\kappa I_{bl}}{U_T}V_{out} + \tfrac{I_{bl}}{U_T}V_{fg} \\ sC_2V_{fg} - s(C_2 + C_L)V_{out} &= \tfrac{\kappa I_{bh}}{U_T}V_{fg} \end{aligned} \tag{63}$$

The sums of capacitances are renamed as

$$C_T = C_1 + C_2 + C_W$$

$$C_O = C_2 + C_L$$

for the total capacitance and the output capacitance, respectively. Again, rewriting the equations, they become

$$\begin{aligned} sC_1V_{in} + sC_2V_{out} - sC_TV_{fg} &= -\tfrac{\kappa I_{bl}}{U_T}V_{out} + \tfrac{I_{bl}}{U_T}V_{fg} \\ sC_2V_{fg} - sC_oV_{out} &= \tfrac{\kappa I_{bh}}{U_T}V_{fg} \end{aligned} \tag{64}$$

By defining two time constants as

$$\tau_l = \frac{C_2U_T}{\kappa I_{\tau_l}}$$

$$\tau_f = \frac{C_2U_T}{\kappa I_{\tau_h}}$$

the equations in (64) become

$$\begin{aligned} s\tfrac{C_1}{C_2}\tau_lV_{in} + s\tau_lV_{out} - s\tfrac{C_T}{C_2}\tau_lV_{fg} &= -V_{out} + \tfrac{1}{\kappa}V_{fg} \\ s\tau_fV_{fg} - s\tfrac{C_O}{C_2}\tau_fV_{out} &= V_{fg} \end{aligned} \tag{65}$$

Rearranging these two expression,

$$\begin{aligned} V_{in}\left(s\tfrac{C_1}{C_2}\tau_l\right) + V_{out}\left(s\tau_l + 1\right) &= V_{fg}\left(s\tfrac{C_T}{C_2}\tau_l + \tfrac{1}{\kappa}\right) \\ V_{fg}\left(s\tau_f - 1\right) &= s\tfrac{C_O}{C_2}\tau_fV_{out} \end{aligned} \tag{66}$$

By multiplying both sides of the first equation in (66) by $(s\tau_f - 1)$, the two equations of (66) can be equated to one another as follows.

$$\begin{aligned} V_{in}\left(s\tfrac{C_1}{C_2}\tau_l\right)\left(s\tau_f - 1\right) + V_{out}\left(s^2\tau_l\tau_f + s\tau_f - s\tau_l - 1\right) = \\ = V_{fg}\left(s\tau_f - 1\right)\left(s\tfrac{C_T}{C_2}\tau_l + \tfrac{1}{\kappa}\right) \\ = V_{out}\left(s\tfrac{C_O}{C_2}\tau_f\right)\left(s\tfrac{C_T}{C_2}\tau_l + \tfrac{1}{\kappa}\right) \\ = V_{out}\left(s^2\tfrac{C_OC_T}{C_2^2}\tau_l\tau_f + s\tfrac{C_O}{\kappa C_2}\tau_f\right) \end{aligned} \tag{67}$$

Solving for the transfer function, the expression becomes

$$\frac{V_{out}}{V_{in}} = -\frac{C_1}{C_2}\frac{s\tau_l(1 - s\tau_f)}{s^2\frac{C_TC_O - C_2^2}{C_2^2}\tau_l\tau_f + s(\tau_l - \tau_f(1 - \frac{C_o}{\kappa C_2})) + 1} \tag{68}$$

However, another time constant can be introduced as

$$\tau_h = \frac{C_T C_O - C_2^2}{C_2} \frac{U_T}{\kappa I_{\tau_h}}$$

This makes the transfer function take its final form as

$$\frac{V_{out}}{V_{in}} = -\frac{C_1}{C_2} \frac{s\tau_l(1 - s\tau_f)}{s^2\tau_h\tau_l + s(\tau_l - \tau_f(1 - \frac{C_o}{\kappa C_2})) + 1} \tag{69}$$

The overall time constant of the filter, which gives the center frequency, is

$$\tau = \sqrt{\tau_l \tau_h}$$

By tuning the filter such that $\tau_h > \tau_l$, resonance occurs, and the value of the Q peak is

$$Q = \sqrt{\frac{\tau_h}{\tau_l}} \frac{1}{1 + \frac{\tau_f}{\tau_l}\left(\frac{C_O}{\kappa C_2} - 1\right)} = \sqrt{\frac{\tau_h}{\tau_l}} \frac{1}{1 + \frac{I_{bl}}{I_{bh}}\left(\frac{C_O}{\kappa C_2} - 1\right)}$$

A.1.2 Maximum Q Peak

As was shown in the previous section, the equation of the Q of the C^4 is given by

$$\begin{aligned} Q &= \sqrt{\frac{\tau_h}{\tau_l}} \frac{1}{1 + \frac{\tau_f}{\tau_l}\left(\frac{C_O}{\kappa C_2} - 1\right)} \\ &= \sqrt{\frac{\tau_h}{\tau_l}} \frac{1}{1 + \frac{I_{bl}}{I_{bh}}\left(\frac{C_O}{\kappa C_2} - 1\right)} \\ &= \sqrt{\frac{C_T C_O - C_2^2}{C_2^2}} \sqrt{\frac{I_{bl}}{I_{bh}}} \frac{1}{1 + \frac{I_{bl}}{I_{bh}}\left(\frac{C_O}{\kappa C_2} - 1\right)} \end{aligned} \tag{70}$$

where the expressions for the time constants were given in the previous section. To find the maximum value of Q, the derivative of (70) needs to be taken. To simplify this derivative, the following substitutions will be made.

$$d = \sqrt{\frac{C_T C_O - C_2^2}{C_2^2}}$$

$$b = I_{bl}/I_{bh}$$

$$a = \frac{C_O}{\kappa C_2} - 1$$

The equation then becomes

$$Q = d\sqrt{b}\frac{1}{1+ba} = d\sqrt{b}\,(1+ba)^{-1}$$

The derivative with respect to b is desired since the current ratio I_{bl}/I_{bh} is the important quantity and the variable within this expression. As the current ratio changes, the value of Q changes. Therefore, the derivative is

$$\begin{aligned}
\frac{dQ}{db} &= d\left(\frac{1}{2}b^{-1/2}\,(1+ab)^{-1} - \sqrt{b}\,(1+ab)^{-2}\,a\right) \\
&= \frac{d}{\sqrt{b}\,(1+ab)^{-2}}\left(\frac{1}{2}\,(1+ab) - ba\right) \\
&= \frac{d}{\sqrt{b}\,(1+ab)^{-2}}\left(\frac{1}{2} - \frac{1}{2}ab\right) \\
&= \frac{d}{2\sqrt{b}\,(1+ab)^{-2}}\,(1-ab) \qquad (71)
\end{aligned}$$

The only feasible root of this equation is when $ab = 1$, which is when $b = 1/a$. By taking the second derivative at this point, this case is shown to be a maximum. Plugging this value of b into the equation for Q, the maximum of the Q peak is shown to be

$$Q_{max} = d\sqrt{\frac{1}{a}}\frac{1}{1+\frac{1}{a}a} = \frac{d}{2\sqrt{a}}$$

By substituting the real values for d and a into this expression, the equation for Q_{max} becomes

$$\begin{aligned}
Q_{max} &= \frac{1}{2}\sqrt{\frac{C_T C_O - C_2^2}{C_2^2}}\frac{1}{\sqrt{\frac{C_O}{\kappa C_2} - 1}} \\
&= \frac{1}{2}\sqrt{\frac{\kappa\,(C_T C_O - C_2^2)}{C_2\,(C_O - \kappa C_2)}} \qquad (72)
\end{aligned}$$

For the case when a high-gain C^4 is used, as with a vanilla C^4, the value of C_2 is very small. This can be used to further approximate the equation for the maximum Q. If C_2 is small, then

$$C_T \approx C_1 + C_W$$

$$C_O \approx C_L$$

This will also mean that $C_T C_O \gg C_2^2$ and $C_O \gg \kappa C_2$. As a result, (72) can be simplified down to

$$Q_{max} \approx \frac{1}{2}\sqrt{\frac{\kappa C_T C_O}{C_2 C_O}} = \frac{1}{2}\sqrt{\frac{\kappa C_T}{C_2}} = \frac{1}{2}\sqrt{\frac{\kappa\left(C_1 + C_W\right)}{C_2}} \tag{73}$$

A.2 Derivation of the C^4 Equations for General Use

When deriving the equations for the C^4, it is appropriate to first look at the operation of the C^4 for the case of widely separated corner frequencies. As a result, the high-frequency operation can be examined separately from the low-frequency operation. Then, the corner frequencies can be brought close together and the case where the C^4 takes on the properties of a bandpass filter with a narrow bandwidth can be examined. In each of these cases, however, the transistors are *not* assumed to necessarily be operating in subthreshold. This general case applies to moderate and strong inversion, as well.

A.2.1 High-Frequency Operation of the C^4

Figure 65 shows the effective operation of the C^4 for the case in which the low corner frequency is much lower than the high corner frequency ($f_l \ll f_h$). This approximation of the C^4 when looking at its high-frequency operation results from the "source-follower" feedback portion being biased at a very low current level. As a result, the current in the source-follower is too small to appreciably charge and discharge the capacitance at the middle node, V_{fg}, at the high frequencies of operation.

The derivation of the transfer function of this high-frequency C^4 case begins by finding the differential equations for the two nodes, which are

$$\begin{aligned} C_1\tfrac{d(V_{in}-V_{fg})}{dt} + C_W\tfrac{d(-V_{fg})}{dt} + C_2\tfrac{d(V_{out}-V_{fg})}{dt} &= 0 \\ C_2\tfrac{d(V_{fg}-V_{out})}{dt} + C_L\tfrac{d(-V_{out})}{dt} &= g_{m4}V_{fg} \end{aligned} \tag{74}$$

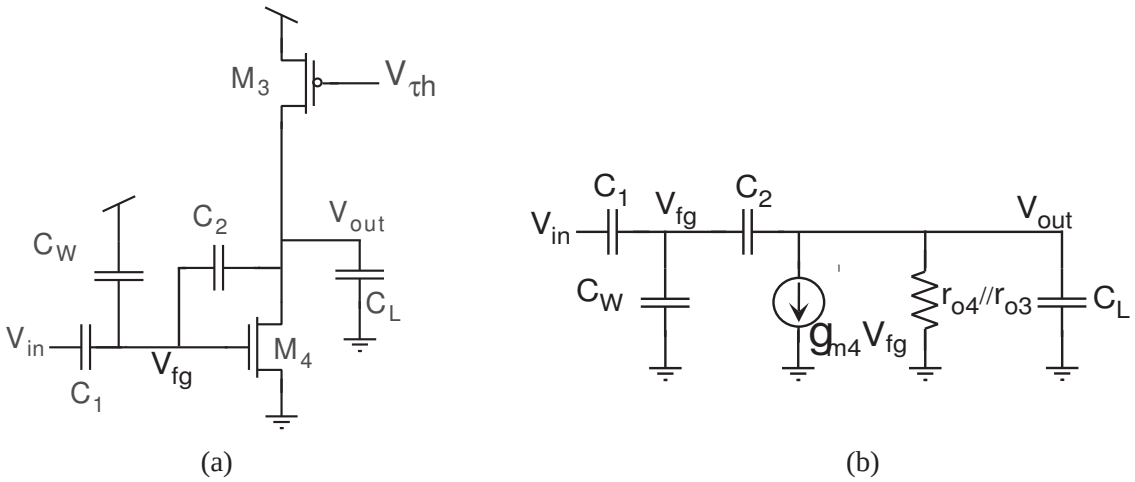


Figure 65. Schematic of the high-frequency properties of the C^4 for widely separated corner frequencies. (a) Equivalent circuit of the C^4 at high frequencies. The feedback loop, consisting of a source-follower, has minimal effect on the circuit response. (b) Small-signal model of the high-frequency equivalent circuit.

By taking the Laplace transform and reorganizing the terms, the equations become

$$\begin{aligned} sC_1V_{in} + sC_2V_{out} &= s(C_1 + C_2 + C_W)V_{fg} \\ (sC_2 - g_{m4})\,V_{fg} &= s(C_2 + C_L)V_{out} \end{aligned} \tag{75}$$

The sums of capacitances are renamed as

$$C_T = C_1 + C_2 + C_W$$

$$C_O = C_2 + C_L$$

for the total capacitance and the output capacitance, respectively. Again, rewriting the equations, they become

$$\begin{aligned} C_1V_{in} + C_2V_{out} &= C_TV_{fg} \\ (sC_2 - g_{m4})\,V_{fg} &= sC_OV_{out} \end{aligned} \tag{76}$$

By defining

$$\tau_f = \frac{C_2}{g_{m4}}$$

as a time constant, the equations can be rewritten as

$$\begin{aligned} C_1 V_{in} + C_2 V_{out} &= C_T V_{fg} \\ (s\tau_f - 1)\, V_{fg} &= s\tfrac{C_O}{C_2} V_{out} \end{aligned} \tag{77}$$

Multiply the top equation of (77) by $(s\tau_f - 1)$, the two equations of (77) can be equated together as follows.

$$\begin{aligned} V_{in}\left(s\frac{C_1}{C_2}\right)(s\tau_f - 1) + V_{out}(s\tau_f - 1) &= V_{fg}(s\tau_f - 1)\left(s\frac{C_T}{C_2}\right) \\ &= V_{out}\left(s\frac{C_T C_O}{C_2^2}\tau_f\right) \end{aligned} \tag{78}$$

Rearranging, (78) becomes

$$\begin{aligned} V_{in}\left(s\frac{C_1}{C_2}\right)(s\tau_f - 1) &= V_{out}\left(s\frac{C_T C_O}{C_2^2}\tau_f - s\tau_f + 1\right) \\ &= V_{out}\left(s\frac{C_T C_O - C_2^2}{C_2^2}\frac{C_2}{g_{m4}} + 1\right) \\ &= V_{out}(s\tau_h + 1) \end{aligned} \tag{79}$$

where

$$\tau_h = \frac{C_T C_O - C_2^2}{C_2 g_{m4}}$$

Therefore, the transfer function is

$$\frac{V_{out}}{V_{in}} = -\frac{C_1}{C_2}\frac{1 - s\tau_f}{1 + s\tau_h} \tag{80}$$

A.2.2 Low-Frequency Operation of the C^4

Figure 66 shows the effective operation of the C^4 for the case in which the high corner frequency is much higher than the low corner frequency ($f_h \gg f_l$). This approximation of the C^4 when looking at its low-frequency operation results from the "common-source amplifier" being biased at a very high current level. As a result, the common-source portion functions as an amplifier over the entire frequency range of interest.

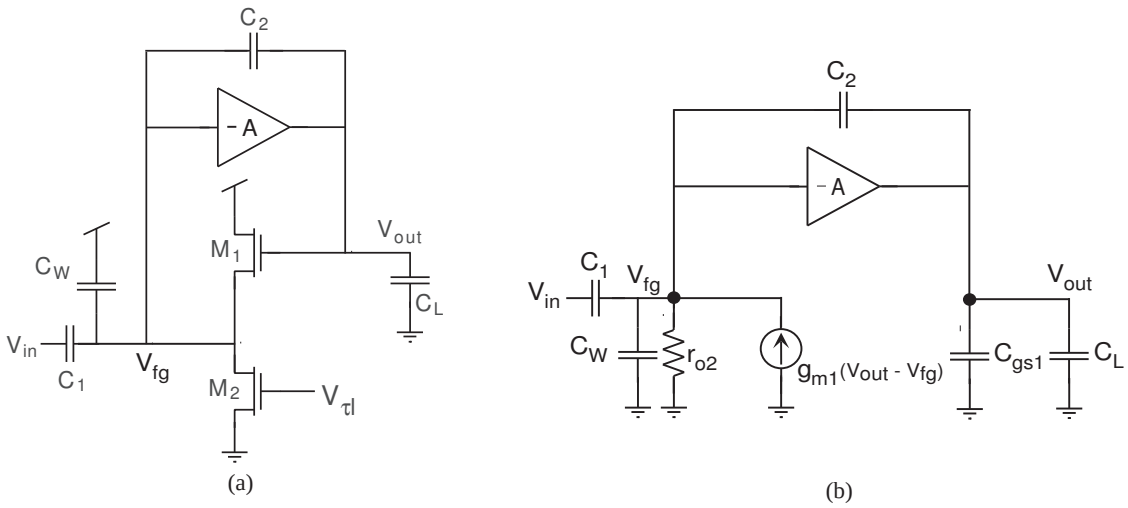


Figure 66. Schematic of the low-frequency properties of the C^4 for widely separated corner frequencies. (a) Equivalent circuit of the C^4 at low frequencies. The common-source amplifier of the C^4 acts as an amplifier with constant gain, A. (b) Small-signal model of the low-frequency equivalent circuit.

The derivation of the transfer function of this low-frequency C^4 case begins by using the Miller effect to replace the capacitor C_2 from the input to output of the high-gain inverting amplifier with a capcitor from V_{fg} to ground of value $C_2(1+A) \approx AC_2$, where A is the gain of the inverting amplifier. The added capacitance at the output node due to the Miller effect is $C_2(1+1/A) \approx C_2$.

The node equation at V_{fg} is, thus,

$$C_1\frac{d(V_{in}-V_{fg})}{dt} + C_W\frac{d(-V_{fg})}{dt} + C_2\frac{d(V_{out}-V_{fg})}{dt} + g_{m1}\left(V_{out}-V_{fg}\right) = 0 \qquad (81)$$

which can be rewritten as

$$sC_1V_{in} + g_{m1}V_{out} = \left(s(C_1+C_2+C_W) + g_{m1}\right)V_{fg} \qquad (82)$$

by using the Laplace transform. By defining the low-frequency time constant as

$$\tau_l = \frac{C_2}{g_{m1}}$$

(82) can be rewritten as

$$V_{in}\left(s\frac{C_1}{C_2}\tau_l\right) + V_{out} = V_{fg}\left(s\frac{C_1+C_W+AC_2}{C_2}\tau_l + 1\right) \qquad (83)$$

The output can be related to V_{fg} by

$$\begin{aligned} V_{out} &= -AV_{fg} \\ V_{fg} &= -\tfrac{V_{out}}{A} \end{aligned} \tag{84}$$

and V_{fg} in (83) can be replaced by (84) so that

$$\begin{aligned} V_{in}\left(s\frac{C_1}{C_2}\tau_l\right) + V_{out} &= -\frac{V_{out}}{A}\left(s\frac{C_1 + C_W + AC_2}{C_2}\tau_l + 1\right) \\ &= -V_{out}\left(s\tau_l\left(\frac{C_1}{AC_2} + \frac{C_W}{AC_2} + \frac{AC_2}{AC_2}\right) + \frac{1}{A}\right) \\ &\approx -V_{out}\left(s\tau_l\left(0 + 0 + 1\right) + 0\right) \\ &\approx -V_{out}\left(s\tau_l\right) \end{aligned} \tag{85}$$

In (85), the gain, A, was assumed to be very large such that $1/A \approx 0$. Finally, the transfer function for the low-frequency case is given by

$$\frac{V_{out}}{V_{in}} = -\frac{C_1}{C_2}\frac{s\tau_l}{1 + s\tau_l} \tag{86}$$

A.2.3 Bandpass Operation of the C^4

The schematic for a C^4 is shown in Figure 67. The derivation of the transfer function of a C^4 begins by finding the differential equations for the two nodes, which are

$$\begin{aligned} C_1\tfrac{d(V_{in} - V_{fg})}{dt} + C_W\tfrac{d(-V_{fg})}{dt} + C_2\tfrac{d(V_{out} - V_{fg})}{dt} &= -g_{m1}\left(V_{out} - V_{fg}\right) \\ C_2\tfrac{d(V_{fg} - V_{out})}{dt} + C_L\tfrac{d(-V_{out})}{dt} &= g_{m4}V_{fg} \end{aligned} \tag{87}$$

By taking the Laplace transforms of (87) and using the following definitions for the time constants

$$\begin{aligned} \tau_l &= \tfrac{C_2}{g_{m1}} \\ \tau_f &= \tfrac{C_2}{g_{m4}} \end{aligned} \tag{88}$$

these equations become

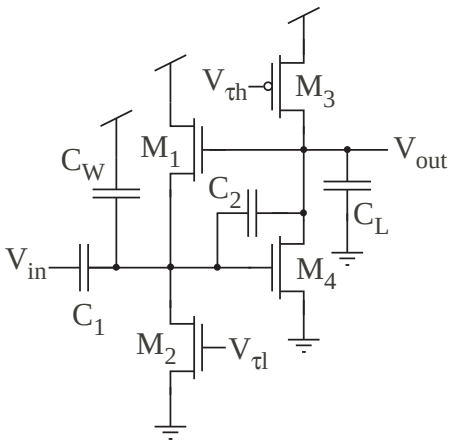


Figure 67. Schematic of the C^4. This is an nFET version of this circuit. A pFET version can also be made by simply flipping the circuit and changing all nFETs to pFETs and all pFETs to nFETs. The overall transfer function is identical for both cases. Only the signs of voltages vary in the initial differential equations.

$$s\frac{C_1}{C_2}\tau_l V_{in} + s\tau_l V_{out} - s\frac{(C_1+C_2+C_W)}{C_2}\tau_l V_{fg} = -V_{out} + V_{fg}$$
$$s\tau_f V_{fg} - s\frac{(C_2+C_L)}{C_2}\tau_f V_{out} = V_{fg} \tag{89}$$

The sums of capacitances are renamed as

$$C_T = C_1 + C_2 + C_W$$
$$C_O = C_2 + C_L$$

for the total capacitance and the output capacitance, respectively. Again, rewriting the equations, they become

$$s\frac{C_1}{C_2}\tau_l V_{in} + s\tau_l V_{out} - s\frac{C_T}{C_2}\tau_l V_{fg} = -V_{out} + V_{fg}$$
$$s\tau_f V_{fg} - s\frac{C_O}{C_2}\tau_f V_{out} = V_{fg} \tag{90}$$

Rearranging these two expression,

$$V_{in}\left(s\frac{C_1}{C_2}\tau_l\right) + V_{out}\left(s\tau_l + 1\right) = V_{fg}\left(s\frac{C_T}{C_2}\tau_l\right)$$
$$V_{fg}\left(s\tau_f - 1\right) = s\frac{C_O}{C_2}\tau_f V_{out} \tag{91}$$

By multiplying both sides of the first equation in (91) by $(s\tau_f - 1)$, the two equations of (91) can be equated to one another as follows.

$$\begin{aligned} V_{in}\left(s\tfrac{C_1}{C_2}\tau_l\right)(s\tau_f - 1) + V_{out}\left(s^2\tau_l\tau_f + s\tau_f - s\tau_l - 1\right) = \\ = V_{fg}(s\tau_f - 1)\left(s\tfrac{C_T}{C_2}\tau_l + 1\right) \\ = V_{out}\left(s\tfrac{C_O}{C_2}\tau_f\right)\left(s\tfrac{C_T}{C_2}\tau_l + 1\right) \\ = V_{out}\left(s^2\tfrac{C_TC_O}{C_2^2}\tau_l\tau_f + s\tfrac{C_O}{C_2}\tau_f\right) \end{aligned} \tag{92}$$

Solving for the transfer function, the expression becomes

$$\frac{V_{out}}{V_{in}} = -\frac{C_1}{C_2}\frac{s\tau_l(1 - s\tau_f)}{s^2\frac{C_TC_O - C_2^2}{C_2^2}\tau_l\tau_f + s\left(\tau_l + \tau_f\left(\frac{C_O}{C_2} - 1\right)\right) + 1} \tag{93}$$

However, another time constant can be introduced as

$$\tau_h = \frac{C_TC_O - C_2^2}{C_2 g_{m4}}$$

This makes the transfer function take its final form as

$$\frac{V_{out}}{V_{in}} = -\frac{C_1}{C_2}\frac{s\tau_l(1 - s\tau_f)}{s^2\tau_h\tau_l + s\left(\tau_l + \tau_f\left(\frac{C_o}{C_2} - 1\right)\right) + 1} \tag{94}$$

The overall time constant of the filter, which gives the center frequency, is

$$\tau = \sqrt{\tau_l\tau_h} = \sqrt{\frac{C_TC_O - C_2^2}{g_{m1}g_{m4}}}$$

By tuning the filter such that $\tau_h > \tau_l$, resonance occurs, and the value of the Q peak is

$$Q = \sqrt{\frac{\tau_h}{\tau_l}}\frac{1}{1 + \frac{\tau_f}{\tau_l}\left(\frac{C_O}{C_2} - 1\right)} = \sqrt{\frac{\tau_h}{\tau_l}}\frac{1}{1 + \frac{g_{m1}}{g_{m4}}\left(\frac{C_O}{C_2} - 1\right)}$$

REFERENCES

[1] C. Mead, *Analog VLSI and Neural Systems*. Reading, MA: Addison-Wesley, 1989.

[2] G. Frantz, "Digital signal processor trends," *IEEE Micro*, vol. 20, pp. 52–59, November-December 2000.

[3] S. Ravindran, P. D. Smith, D. W. Graham, V. Duangudom, D. V. Anderson, and P. Hasler, "Towards low-power on-chip auditory processing," *EURASIP Journal on Applied Signal Processing*, vol. 2005, pp. 1082–1092, May 2005.

[4] J. Glossner, D. Routenberg, E. Hokenek, M. Moudgill, M. J. Schulte, P. I. Balzola, and S. Vassiliadis, "Towards a very high bandwidth wireless battery powered device," *IEEE Computer Society Workshop on VLSI*, pp. 3–9, April 2001.

[5] P. Hasler, P. D. Smith, D. W. Graham, R. Ellis, and D. V. Anderson, "Analog floating-gate, on-chip auditory sensing system interfaces," *IEEE Sensors Journal*, vol. 5, pp. 1027–1034, October 2005.

[6] R. Sarpeshkar, *Efficient Precise Computation with Noisy Components: Extrapolating from an Electronic Cochlea to the Brain*. PhD thesis, California Institute of Technology, Pasadena, CA, 1997.

[7] S. Y. Peng, M. S. Qureshi, P. Hasler, N. Hall, and F. Degertekin, "High SNR capacitive sensing transducer," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, 2006, In Press.

[8] R. Ellis, H. Yoo, D. W. Graham, P. Hasler, and D. V. Anderson, "A continuous-time speech enhancement front-end for microphone inputs," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 2, (Scottsdale, AZ), pp. II–728 – II–731, May 2002.

[9] P. D. Smith, M. Kucic, R. Ellis, P. Hasler, and D. V. Anderson, "Mel-frequency cepstrum encoding in analog floating-gate circuitry," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 4, (Scottsdale, AZ), pp. IV–671 – IV–674, May 2002.

[10] T. Massengill, D. Wilson, P. Hasler, and D. W. Graham, "Empirical comparison of analog and digital auditory preprocessing for automatic speech recognition," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 5, (Scottsdale, AZ), pp. V–77 – V–80, May 2002.

[11] L. Watts, *Cochlear Mechanics: Analysis and Analog VLSI*. PhD thesis, California Institute of Technology, Pasadena, CA, 1993.

[12] A. van Schaik, E. Fragnière, and E. Vittoz, "Improved silicon cochlea using compatible lateral bipolar transistors," in *Advances in Neural Information Processing Systems 8* (D. Touretzky, ed.), (Cambridge, MA), pp. 671–677, MIT Press, 1996.

[13] A. van Schaik and E. Fragnière, "Pseudo-voltage domain implementation of a 2-dimensional silicon cochlea," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 2, (Sydney, NSW, Australia), pp. 185–188, May 2001.

[14] R. Sarpeshkar, R. F. Lyon, and C. A. Mead, "An analog VLSI cochlea with new transconductance amplifiers and nonlinear gain control," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 3, (Atlanta, GA), pp. 292–296, 1996.

[15] H. Shiraishi and A. van Schaik, "A 2-dimensional active cochlear model for analog vlsi implementation," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 4, (Vancouver, BC, Canada), pp. IV–924 – IV–927, May 2004.

[16] C. Salthouse and R. Sarpeshkar, "A practical micropower programmable bandpass filter for use in bionic ears," *IEEE Journal of Solid State Circuits*, vol. 38, pp. 63–70, January 2003.

[17] E. Kandel, J. Schwartz, and T. Jessel, *Principles of Neural Science*. New York: McGraw-Hill, 4th ed., 2000.

[18] R. Berne and M. Levy, *Physiology*. New York: Mosby, 4th ed., 1998.

[19] P. Denes and E. Pinson, *The Speech Chain*. Freeman, 2nd ed., 1993.

[20] C. Mead and R. Lyon, "An analog electronic cochlea," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 36, pp. 1119–1134, July 1988.

[21] T. Gold and R. J. Pumphrey, "Hearing. I. The cochlea as a frequency analyzer," *Proceedings of the Royal Society of London. Series B, Biological Sciences*, vol. 135, pp. 462–491, December 1948.

[22] W. S. Rhode, "Observations of the vibration of the basilar membrane in squirrel monkeys using the Mössbauer technique," *Journal of the Acoustical Society of America*, vol. 49, pp. 1218–1231, April 1971.

[23] D. W. Graham and P. Hasler, "Capacitively-coupled current conveyer second-order sections for continuous-time bandpass filtering and cochlea modeling," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 5, (Scottsdale, AZ), pp. V–485 – V–488, May 2002.

[24] E. Fragnière, "100-channel analog CMOS auditory filter bank for speech recognition," in *Digest of Technical Papers from the IEEE International Solid-State Circuits Conference*, vol. 3, (San Francisco, CA), pp. 140–141, February 2005.

[25] B. Wen and K. Boahen, "A linear cochlear model with active bi-directional coupling," in *Proceedings of the 25th Annual International Conference of the IEEE Engineering in Medecine and Biology Society*, vol. 3, pp. 2013–2016, September 2003.

[26] B. Wen and K. Boahen, "A 360-channel speech preprocessor that emulates the cochlear amplifier," in *Digest of Technical Papers from the IEEE International Solid-State Circuits Conference*, pp. 556–557, February 2006.

[27] B. Wen and K. Boahen, "Active bidirectional coupling in a cochlear chip," in *Advances in Neural Information Processing Systems 9* (B. Sholkopf and Y. Weiss, eds.), (Cambridge, MA), MIT Press, 2006.

[28] P. Hasler, M. Kucic, and B. A. Minch, "A transistor-only circuit model of the autozeroing floating-gate amplifier," in *IEEE Midwest Symposium on Circuits and Systems*, (Las Cruces), pp. 157–160, August 1999.

[29] M. Kucic, A. Low, P. Hasler, and J. Neff, "Programmable continuous-time floating-gate Fourier processor," *IEEE Transactions on Circuits and Systems II*, vol. 48, pp. 90–99, January 2001.

[30] P. Hasler, B. A. Minch, C. Diorio, and C. A. Mead, "An autozeroing floating-gate amplifier," *IEEE Transactions on Circuits and Systems II*, vol. 48, no. 1, pp. 74–82, 2001.

[31] D. W. Graham, "Continuous-time bandpass second-order sections and their applications in cochlea modeling," Master's thesis, Georgia Institute of Technology, Atlanta, GA, 2003.

[32] P. D. Smith, D. W. Graham, R. Chawla, and P. Hasler, "A five-transistor bandpass rilter element," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 1, (Vancouver, BC, Canada), pp. I–861 I–864, May 2004.

[33] L. Watts, D. A. Kerns, R. F. Lyon, and C. Mead, "Improved implementation of the silicon cochlea," *IEEE Journal of Solid-State Circuits*, vol. 27, pp. 692–700, May 1992.

[34] R. Chawla, D. W. Graham, P. D. Smith, and P. Hasler, "A low-power, programmable bandpass filter section for higher-order filter-bank applications," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, (Kobe, Japan), pp. 1980 – 1983, May 2005.

[35] R. Chawla, D. W. Graham, P. D. Smith, and P. Hasler, "A low-power, programmable bandpass filter section for higher-order filter applications," *IEEE Transactions on Circuits and Systems I*, Submitted.

[36] H. J. Yoo, D. W. Graham, D. V. Anderson, and P. Hasler, "C^4 bandpass delay filter for continuous-time subband adaptive tapped-delay filter," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 5, (Vancouver, BC, Canada), pp. V–792 – V–795, May 2004.

[37] R. Chawla, G. Serrano, D. Allen, A. Periera, and P. Hasler, "Fully differential floating-gate programmable otas with novel common-mode feedback," in *Proceedings of the International Symposium on Circuits and Systems*, vol. 1, (Vancouver, BC, Canada), pp. 817–820, May 2004.

[38] P. Hasler, B. A. Minch, C. Diorio, and C. Mead, "An autozeroing amplifier using pFET hot-electron injection," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. III, (Atlanta, GA), pp. 325–328, 1996.

[39] P. Hasler, B. A. Minch, and C. Diorio, "An autozeroing floating-gate bandpass filter," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, (Monterey, CA), pp. 131–134, 1998.

[40] R. Sarpeshkar, R. F. Lyon, and C. Mead, "A low-power wide-linear-range transconductance amplifier," *Analog Integrated Circuits and Signal Processing*, vol. 13, no. 1-2, pp. 123–151, 1997.

[41] P. Hasler, T. Stanford, B. A. Minch, and C. Diorio, "An autozeroing floating-gate second-order section," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 2, (Monterey, CA), pp. 351–354, May 1998.

[42] P. Hasler, P. D. Smith, R. Ellis, D. W. Graham, and D. V. Anderson, "Biologically inspired sensing system interfaces on a chip," in *Proceedings of the IEEE Sensors*, vol. 1, (Orlando, FL), pp. 669–674, June 2002.

[43] P. D. Smith, M. Kucic, and P. Hasler, "Accurate programming of analog floating-gate arrays," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 5, (Scottsdale, AZ), pp. V–489 – V–492, May 2002.

[44] P. Hasler, B. A. Minch, and C. Diorio, "Adaptive circuits using pFET floating-gate devices," in *Proceedings of the 20th Anniversary Conference on Advanced Research in VLSI*, (Atlanta, GA), pp. 215–229, March 1999.

[45] A. Bandyopadhyay, G. Serrano, and P. Hasler, "Programmaing analog computational elements to 0.2% accuracy over 3.5 decades using a predictive method," in *Proceedings of the International Symposium on Circuits and Systems*, vol. 3, (Kobe, Japan), pp. 2148–2151, May 2005.

[46] V. Srinivasan, D. W. Graham, and P. Hasler, "Floating-gate transistors for precision analog circuit design: An overview," in *Midwest Symposium on Circuits and Systems*, vol. 1, (Covington, KY), pp. 71–74, August 2005.

[47] E. Sackinger and W. Guggenbuhl, "An analog trimming circuit based on a floating-gate device," *Journal of Solid State Circuits*, vol. 23, pp. 1437–1440, December 1988.

[48] H. Nozama and S. Kokyama, "A thermionic electron emission model for charge retention in SAMOS structures," *Japanese Journal of Applied Physics*, vol. 21, pp. L111–L112, February 1992.

[49] C. Bleiker and H. Melchior, "A four-state EEPROM using floating-gate memory cell," *Journal of Solid State Circuits*, vol. 22, pp. 460–463, June 1987.

[50] L. R. Carley, "Trimming analog circuits using floating-gate analog MOS memory," *Journal of Solid State Circuits*, vol. 24, pp. 1569–1574, December 1989.

[51] D. W. Graham, E. Farquhar, B. Degnan, C. Gordon, and P. Hasler, "Indirect programming of floating-gate transistors," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, (Kobe, Japan), pp. 2172 – 2175, May 2005.

[52] D. W. Graham, E. Farquhar, B. Degnan, C. Gordon, and P. Hasler, "Indirect programming of floating-gate transistors," *IEEE Transactions on Circuits and Systems I*, Submitted.

[53] G. Serrano, P. D. Smith, H. J. Lo, R. Chawla, T. S. Hall, C. M. Twigg, and P. Hasler, "Automatic rapid programming of large arrays of floating-gate elements," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 1, (Vancouver, BC, Canada), pp. 1373–1376, May 2004.

[54] F. Adil, G. Serrano, and P. Hasler, "Offset removal using floating-gate circuits for mixed-signal systems," in *Proceedings of the IEEE Southwest Symposium on Mixed-Signal Design*, (Las Vegas, NV), pp. 190–195, February 2003.

[55] P. Hasler, A. G. Andreou, C. Diorio, B. A. Minch, and C. A. Mead, "Impact ionization and hot-electron injection derived consistently from boltzmann transport," *VLSI Design*, vol. 8, no. 1-4, pp. 455–461, 1998.

[56] P. Kinget, "Device mismatch and tradeoffs in the design of analog circuits," *IEEE Journal of Solid-State Circuits*, vol. 40, no. 6, pp. 1212–1224, 2004.

[57] K. Yang and A. G. Andreou, "Subthreshold analysis of floating-gate MOSFET's," in *Proceedings of the Tenth Biennial University/Government/Industry Microelectronics Symposium*, (Research Triangle Park, NC), pp. 141–144, May 1993.

[58] J. Gray, C. Twigg, D. Abramson, and P. Hasler, "Characteristics and programming of floating-gate pFET switches in an FPAA crossbar network," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 1, (Kobe, Japan), pp. 468–471, May 2005.

[59] B. Degnan, R. Wunderlich, and P. Hasler, "Programmable floating-gate techniques for CMOS inverters," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 3, (Kobe, Japan), pp. 2441–2444, May 2005.

[60] R. Harrison, J. Bragg, P. Hasler, B. Minch, and S. Deweerth, "A CMOS programmable analog memory cell array using floating-gate circuits," *IEEE Transactions on Circuits and Systems II*, vol. 48, pp. 4–11, January 2001.

[61] P. Hasler and J. Dugger, "An analog floating-gate node for supervised learning," *IEEE Transactions on Circuits and Systems I*, vol. 52, pp. 834–845, May 2005.

[62] A. Low and P. Hasler, "Basics of floating-gate low-dropout voltage regulators," in *Proceedings of the IEEE Midwest Symposium on Circuits and Systems*, vol. 3, pp. 1048–1051, August 2000.

[63] E. Farquhar and P. Hasler, "A bio-physcially inspired silicon neuron," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 1, (Vancouver, BC, Canada), pp. I–309 – I–312, May 2003.

[64] D. W. Graham, P. D. Smith, R. Chawla, and P. Hasler, "A programmable bandpass array using floating-gate elements," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 1, (Vancouver, BC, Canada), pp. I–97 I–100, May 2004.

[65] D. W. Graham, P. D. Smith, and P. Hasler, "A programmable array of bandpass filters using floating-gate transistors for smart audio sensors," *IEEE Transactions on Circuits and Systems I*, Submitted.

[66] W. S. Rhode, *Measurement of the Amplitude and Phase of Vibration of the Basilar Membrane using the Mössbauer Effect*. PhD thesis, University of Wisconsin, 1970.

[67] M. C. Holley and J. F. Ashmore, "A cytoskeletal spring in cochlear outer hair cells," *Nature*, vol. 335, pp. 635–637, October 1988.

[68] H. J. Yoo, D. V. Anderson, and P. Hasler, "Continuous-time audio suppression and real-time implementation," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. IV, (Orlando, FL), pp. 3980–3983, May 2002.

[69] D. V. Anderson, P. Hasler, R. Ellis, H. J. Yoo, D. W. Graham, and M. Hans, "A low-power system for audio noise suppression: A cooperative analog-digital signal processing approach," in *Proceedings of the IEEE Digital Signal Processing Workshop*, (Pine Mtn., GA), pp. 327 – 332, October 2002.

[70] J. R. Deller, J. G. Proakis, and J. H. L. Hansen, *Discrete-Time Processing of Speech Signals*. New York: MacMillan, 1993.

[71] P. D. Smith, D. W. Graham, and P. Hasler, "Low-power speech processing based upon floating-gate circuits," in *Conference Records of the 37th Asilomar Conference on Signals, Systems, and Computers*, vol. 2, (Pacific Grove, CA), pp. 2026 – 2030, November 2003.

[72] X. Yang, K. Wang, and S. Shamma, "Auditory representations of acoustic signals," *IEEE Transactions on Information Theory*, vol. 38, pp. 824–839, March 1992.

[73] K. Wang and S. Shamma, "Self-normalization and noise-robustness in early auditory representations," *IEEE Transactions on Speech and Audio Processing*, vol. 2, pp. 421–435, July 1994.

[74] T. Hall, C. Twigg, J. Gray, P. Hasler, and D. V. Anderson, "Large-scale field-programmable analog arrays for analog signal processing," *IEEE Transactions on Circuits and Systems I*, vol. 52, pp. 2298–2307, November 2005.

[75] F. Baskaya, S. Reddy, S. K. Lim, and D. V. Anderson, "Offset removal using floating-gate circuits for mixed-signal systems," in *Proceedings of the International Conference on Field Programmable Logic and Applications*, pp. 421–426, August 2005.

[76] E. Farquhar, D. Abramson, and P. Hasler, "A reconfigurable bidirectional active 2 dimensional dendrite model," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 1, (Vancouver, BC, Canada), pp. I–317 – I–320, May 2004.

[77] D. Abramson, J. Gray, S. Subramanian, and P. Hasler, "A field-programmable analog array using translinear elements," in *Proceedings of the Fifth International System-On-Chip for Real-Time Applications*, pp. 425 – 428, July 2005.