A Cochlear Filter Implemented With a Field-Programmable Gate Array

Apurva Mishra and Allyn E. Hubbard

***Abstract*—A digital cochlear filter was designed and implemented in hardware using a Xilinx XC4010 field-programmable gate array (FPGA) chip. The filter gives a good fit to biological data. It is a tenth-order recursive filter implemented as a parallel combination of low-order elements. A systematic approach to the design problem enabled us to achieve a good fit to cochlear data with efficiency of hardware usage. This approach can be easily extended to other similar applications.**

***Index Terms*—Cochlea, field-programmable gate array (FPGA), infinite-impulse response (IIR) filter.**

I. Introduction

THE motivation for developing an electronic model of the cochlea is the potential use in hearing aids, prosthetic implants, and systems that aim to extract information from sound signals in a biomimetic fashion. The processing that takes place in the biological cochlea (up to but excluding the action of the hair cells) is usually modeled as an array of approximately low-pass filters, having extraordinary filter shapes around their cutoff frequencies, and whose cutoff frequencies range over the entire audible spectrum.

To implement a cochlear filter bank in hardware, a digital implementation is potentially suitable. The digital embodiment is robust and reliable as compared with analog implementations, which have been typically carried out using a design style called subthreshold analog design, typically designated aVLSI, although the "a" does not necessarily denote subthreshold design methodology. The first cochlear implementation using aVLSI circuitry was carried out by Mead and Lyon [1]. This work has been the genesis of numerous, subsequent analog implementations of cochlear filters, the most recent of which has been done by Xing [2] in our laboratory.

Although touted for their low power consumption, aVLSI subthreshold circuits are fraught with difficulty due to variations in process and temperature. The process variations may change the exponential coefficient governing the current-voltage relationship by a factor of two. Moreover, the basic currency of the exponential is 25 mV (kT/q), each unit of which changes currents by a factor of e. Moreover, these designs can absolutely fail to function when driven outside a limited input range.

In spite of these difficulties that plague aVLSI, the amount of work done so far in developing digital implementations has been scanty. Kates [3] carried out a digital implementation in software using a cascade of third-order filters for the initial filtering followed by a bank of second-order filters. A similar implementation in hardware was achieved by Summerfield and Lyon [4]. In this embodiment, the filters used were now second-order and first-order, respectively. Brucke *et al.* put forth an implementation involving a bank of gammatone filters [5]. The design was apparently submitted for fabrication, but test results of the actual hardware have not been presented. Lim *et al.* designed using VHDL a model using first-order Butterworth band-pass filters [6]. The hardware test of this design has also not been reported.

All of these implementations are fairly simplistic and in most cases even their target performance compares poorly with biological data. In this paper we present a digital implementation that mimics the performance measured from a single tap of Hubbard's analog traveling-wave amplifier model (TWAmp model)[7]. The TWAmp model provides an excellent fit to the best available physiological data measured from the mammalian cochlea. High, but not narrow peaks characterize the data. In other words, the high peaks are not high-Q, but they are relatively broad. Moreover, the phase is nearly constant up to the tuned frequency, and then it drops precipitously. This is the desired performance of a cochlear filter.

Our digital filter hardware implementation mimics well the performance of high-frequency cochlear filters. More importantly, we present a methodology that can be useful for future cochlear filter design, for example, those filters tuned to lower frequency. Low frequency cochlear filters have decidedly different characteristics from those tuned to high frequencies.

We used a 10-pole filter with 8-bit coefficients to implement the desired transfer function with a high degree of accuracy. We implemented and tested our design on a Xilinx XC4010XL field-programmable gate array (FPGA) chip. The flexibility inherent in the FPGA design process allowed us to use differently scaled coefficients and arbitrary internal wordlengths useful to our application, which would not be possible with a digital signal processor (DSP) chip. As our immediate motivation was use in biomimetic systems for extracting information from sound for defense and industrial applications, the disadvantages of FPGA *vis-à-vis* application specific integrated circuit (ASIC) implementation in terms of size and power consumption were not of concern and more than compensated for by the advantages of cost and flexibility. The same filter

Manuscript received August 21, 2000; revised January 7, 2002. This work was supported by the Office of Naval Research under a Multidisciplinary University Research Initiatives Grant MURI N00014-97-1-0501. This paper was recommended by Associate Editor T. S. Lande.

A. Mishra was with the Department of Electrical and Computer Engineering, Boston University, Boston, MA 02215 USA. He is now with the Desktop Platforms Group, Intel Corporation, Hillsboro, OR 97124 USA (e-mail: apurva.mishra@intel.com).

A. E. Hubbard is with the Departments of Electrical and Computer Engineering and Biomedical Engineering, Boston University, Boston, MA 02215 USA (e-mail: aeh@bu.edu).

Publisher Item Identifier S 1057-7130(02)02785-4.

design could however be implemented on an ASIC for use in prosthetic devices.

Section II of this paper describes in detail filter design and architecture issues. Section III describes how the circuit was implemented on the Xilinx chip. Section IV demonstrates the results and Section V concludes the paper.

II. Filter Design

A. Design Options

There are a large number of options available for filter design and implementation. We have systematically chosen options suitable to our application. Thus, we have achieved a good fit to the targeted performance, while maintaining low complexity in hardware.

The target chosen for the filter design was the frequency response of Hubbard's TWAmp model [7] of the cochlea. Typically, the design target for a filter may be specified in several ways. For low-pass, bandpass, or high-pass filters, it can be specified in terms of bandpass and bandstop frequencies and gain levels. However the cochlear filter function does not fall into any of these special categories. Therefore a least mean square error (LMSE) approach is well suitable for our need, and accordingly we used an LMSE design algorithm to construct the filter from the desired transfer function. The Matlab function *invfreqz* was used for this purpose [8].

Another issue is whether to use a finite impulse response (FIR) or infinite impulse response (IIR) filter to implement the desired transfer function. Implementing a FIR filter is typically easier than implementing an IIR filter. It is usually considered a virtue that an FIR filter can have linear phase, while an IIR filter cannot. But our target filter characteristic does not have a linear phase response, so this benefit of FIR filters is useless to us. Importantly, an IIR filter can achieve the high and narrow passband required of a cochlear filter with far fewer coefficients (and consequently fewer arithmetic operations) than an FIR filter. Therefore, an IIR filter implementation was chosen.

The next step was choosing the number of coefficients. The choice of order is somewhat subjective; and was decided based on the intention of arriving at a closer fit to cochlear data than previously achieved, with the premise that the present-day cost of digital hardware can allow for many high-order filters to be implemented on an array of FPGAs with relatively low cost. Put differently, it is more valuable to create larger high-order filters that closely match the cochlea, than smaller low-order ones that fail to match cochlear data well. Closely matching cochlear response is critical for subsequent application of biomimetic post-cochlea processing, because all of these mechanisms presuppose a "normal" cochlea is attached. It is difficult to achieve the high passband and sharp cutoff with a lower order filter. Accordingly we decided on a 10-zero 20-pole filter, which achieves a significantly accurate match to cochlear data, and is not prohibitively complicated to implement. The coefficients are listed in Table I.

Fig. 1 depicts the pole-zero plot of the filter design, with unquantized coefficients. The unquantized coefficients for our design are such that all poles are within the unit circle, but very close to it. When we quantize coefficients, the underlying polynomial representation of the filter changes; and it is possible that this might throw poles outside the unit circle. Instability results when a pole moves outside the unit circle. Even if they do not actually leave the unit circle, small changes in pole locations can result in large changes in filter response.

TABLE I
Filter Coefficients for Overall Filter

Coefficient	Value
b_0	-0.156422
b_1	0.491060
b_2	-0.221332
b_3	1.063537
b_4	-0.630294
b_5	1.291993
b_6	-0.023245
b_7	0.894643
b_8	0.473989
b_9	0.092871
b_{10}	-0.235991
b_{11}	-0.997991
b_{12}	-1.194547
b_{13}	-1.241448
b_{14}	-0.645134
b_{15}	-0.180764
b_{16}	0.265412
b_{17}	0.259285
b_{18}	0.252364
b_{19}	0.074707
b_{20}	0.060968
a_0	1.000000
a_1	-4.731289
a_2	13.258728
a_3	-25.001087
a_4	35.255925
a_5	-37.788058
a_6	31.403429
a_7	-19.799585
a_8	9.245326
a_9	-2.887853
a_{10}	0.505565

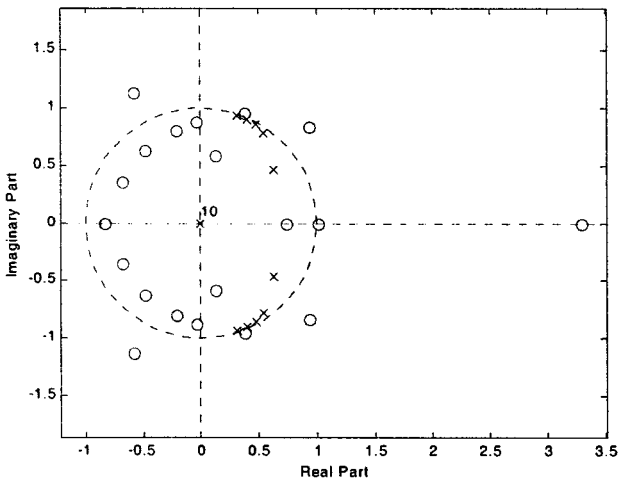

Fig. 1. Pole-zero plot of IIR design.

The actual implementation of an IIR design can be done in several different ways. Some of the commonly used implementations are the direct form (DF), the cascade, and the parallel forms. While these different architectures are identical in the case of infinite-precision computation and storage, they differ significantly when using finite-precision coefficients [9]. The DF is not usually advisable for high-order IIR filters owing to its high sensitivity to coefficient quantization, which can lead, among other things, to severe stability problems, especially for the high-gain, tuned responses that we are targeting. As can be seen from Fig. 1, the poles are clustered close to each other and to the unit circle, making it highly sensitive to coefficient quantization effects. The ten trivial poles shown at the origin are not relevant in any way and may be ignored.

The cascade and parallel forms are based on decomposing the high-order transfer function into a cascade or parallel combination of lower-order (usually second-order) filters, and are significantly less sensitive to coefficient quantization. Thus, it is desirable to use such a combination rather than a high order DF implementation.

Next arises the question of choosing between the cascade and parallel forms. It is generally beneficial that implementation using the cascade form allows choosing from among a large number of possible permutations in implementation. Also, the cascade form allows for easy introduction of zeroes on the unit circle [10]. However, since our desired high-order response did not have any zeroes exactly on the unit circle, this benefit of the cascade form is of no value to us. Thus, we chose to implement the parallel form.

The parallel form has many advantages. It typically has a noise level comparable to that of the best possible cascade form for the same overall response [11]. A block diagram of the filter implemented as a combination of lower order IIR filters and one FIR filter is shown in Fig. 2. The single FIR filter in the setup serves to implement the remaining zeroes after the poles have been paired up.

At the lowest level, we are using second-order recursive filters (also called biquads). There are several ways to implement a biquad: direct form 1, direct form 2, transposed form 1, and transposed form 2. As explained in [12], Direct form 1 is most

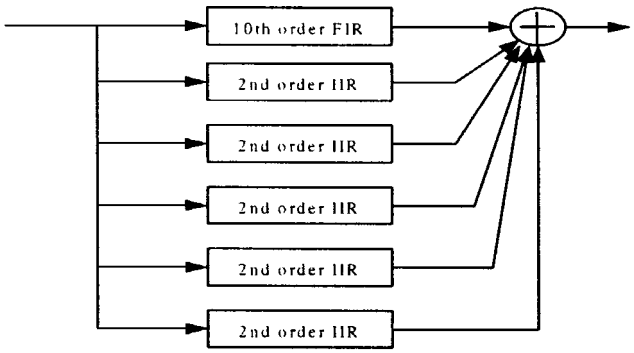

Fig. 2. Block diagram of IIR design.

suitable when we do not have the need to truncate the arithmetic result at intermediate stages, and we know exactly the maximum required word length at every stage, and we can set the word length so that truncation will not occur. These factors considered, we chose direct form 1 to implement the biquads.

Each of biquads has a transfer function of the form

$$\boldsymbol{H}(z) = \frac{e_0 + e_1 z^{-1}}{\mathbf{1} - a_1 z^{-1} - a_2 z^{-2}}$$

and the single component FIR block has a transfer function of the form

$$\boldsymbol{H}(z) = \sum_{n=0}^{10} k_n z^{-n}.$$

The coefficients are listed in Table II(a) and (b).

By making systematic decisions at each stage of the design process, we arrived at an architecture that was suitable for implementing the target cochlea-like response.

B. Architecture Details

Several word lengths must be considered. We chose an input word width of eight bits, because data for our target application are eight bits wide. The cochlear filter has very high gain (on the order of 2^{13}). It is therefore necessary to have an internal wordlength that accommodates the maximum possible output (that is, maximum possible input multiplied by maximum possible gain), to maintain the input precision. Thus, for an input word of eights bits, we used an internal wordlength of $8 + 13 = 21$ bits.

As with any recursive filter, here too it was necessary to build in safeguards against overflow that may cause erroneous results and can produce instability. This was accomplished by building in a saturating nonlinearity as described in [12] that clips the output of the accumulator to the maximum or minimum possible value rather than allowing wraparound to occur. The saturation was designed not to take place until the last accumulation had been performed, since so long as the final sum lies within the desired numerical range, it is acceptable for the partial sums to exceed it [9]. The saturation is implemented by using some redundant bits at the most significant bit (MSB) end of the accumulator word and testing them after the last accumulation has been performed to determine whether the sum lies outside the range of the output word. If the sum lies outside the range,

TABLE II
FILTER COEFFICIENTS FOR COMPONENT BIQUADS

Coefficient	Biquad 1	Biquad 2	Biquad 3	Biquad 4	Biquad 5
e_0	-5.8603	-77.6106	100.9273	110.1580	-86.9512
e_1	15.3999	54.4944	-293.0241	162.6557	157.1701
a_1	0.6241	0.8080	0.9551	1.0849	1.2591
a_2	-0.9753	-0.9730	-0.9579	-0.9019	-0.6167

(a)

Coefficient	Value
k_0	-40.8196
k_1	-137.0095
k_2	-117.2149
k_3	-56.8831
k_4	-9.4080
k_5	10.8128
k_6	12.3813
k_7	7.4883
k_8	3.0727
k_9	0.8366
k_{10}	0.1206

(b)

it is clipped to the maximum or minimum value of the range based on whether the redundant bits form a positive or negative number. Note that this care is taken only for the first accumulator (internal to the second-order filters). This is to minimize error in signals that are being fed back. No such action was taken for the second accumulator (which adds up the outputs of the low-order sections) since its output is not being fed back.

Note that the abovementioned saturation is intended solely as a safeguard against the possibility of certain unusual sequences of input causing wraparound. Most of the time, it will not come into play.

We were able to use eight bits for representing the coefficients and still achieve a good fit to the target filter response. This was possible because we scaled the coefficient values to take maximum advantage of the eight bits. Our need to do this stems from the fact that for each biquad there is a large difference of scale among the four coefficients. Therefore, a scaling algorithm was used for representing the coefficient values. An inverse scaling takes place after the multiply operation. This allows us to reduce coefficient wordlength without compromising precision.

Simulations of filter response using our 8-bit coefficients were carried out. The plots are based on assumption of a sampling frequency of 24 kHz. As seen in Fig. 3, the simulated response fits the target well in spite of the fact that poles are clustered close to each other and to the unit circle. For practical reasons related to the application of the digital filter in acoustic environments, the differences in performance do not outweigh the utility of the more compact hardware implementation.

III. IMPLEMENTATION ON XILINX CHIP

Since the input signal is audio frequency, and the Xilinx XC4010 FPGA can be operated up to 50 MHz, it is possible to serialize the multiplication and addition operations for greater area-efficiency. Thus, we use a single multiplier and accumulator to perform all the operations. For this we used the following:

- an internal clock that is faster than the input signal clock speed by a factor of at least N, where N is the number of coefficient multiply/adds required;
- a multiplexer that sends the appropriate signal to the multiply-accumulate;
- a single shift register carrying the input signal values (feedforward signal values), and several shift registers,

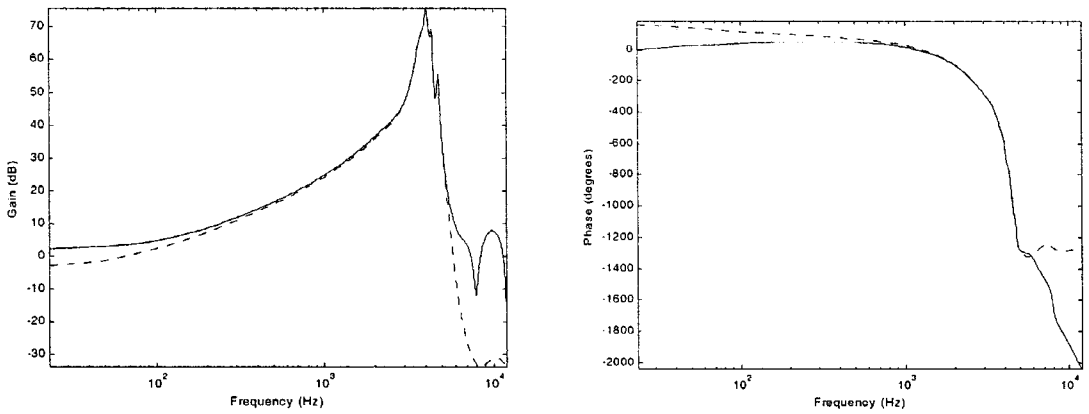

Fig. 3. Software simulations of magnitude and phase response. The dotted lines represent simulations with infinite-precision coefficients. The solid lines are for 8-bit quantized coefficients.

one for each of the biquads, carrying the output signal values (feedback signal values);
- two accumulators, one for computing the output of the lower-order sections, and one for summing up these outputs to get the final output;
- a ROM block for 8-bit coefficient storage;
- a scaling block for undoing the effect of scaling used in representation of coefficient values;
- generation of appropriate control signals.

The blocks were written in VHDL. The Xilinx Foundation software was used for design entry, synthesis, and simulation.

The present design takes nearly 100% of the resources present in the XC4010 Xilinx chip. More details are provided in Table III. We have not spent any effort on reducing further the FPGA resource usage by specifying low-level descriptions of the different functional blocks, such as the multiplier and accumulator, or by using advanced optimization tools. Our entire focus concerning circuit size was to reduce complexity at a high level by choice of design and architecture.

TABLE III
XILINX CHIP RESOURCE USAGE

Number of CLBs:	400 out of 400 100%
CLB Flip Flops:	445
CLB Latches:	0
4 input LUTs:	751 (2 used as route-throughs)
3 input LUTs:	219 (134 used as route-throughs)
32X1 ROMs:	16
Number of bonded IOBs:	31 out of 65 47%
IOB Flops:	25
IOB Latches:	0
Number of clock IOB pads:	1 out of 12 8%
Number of TBUFs:	16 out of 880 1%
Number of BUFGLSs:	1 out of 8 12%
65 unrelated functions packed into 65 CLBs.	
(16% of the CLBs used are affected.)	
Total equivalent gate count for design: 10261	
Additional JTAG gate count for IOBs:	1488

IV. RESULTS

We tested the circuit in hardware using sinusoidal inputs with varying frequencies and amplitudes. A sampling frequency of 48 kHz was used. Fig. 4 depicts the hardware test results in the form of amplitude and phase response curves. The hardware test result was compared with the quantized-coefficient software simulations (after adjusting the latter for the difference in sampling frequency). So close is the fit of the hardware test result to the quantized simulation data, that one can scarcely tell that two lines are plotted rather than one. The biological data from Ruggero *et al.* [13] are also shown.

Fig. 5 depicts the filter response for a variety of different input amplitudes. This particular version of the filter can neither saturate nor overflow. Thus, the high level input situation has the best signal to noise ratio. Response curves for very low input amplitudes appear fuzzy owing to the quantization noise present at the input itself, which is not a failing of the filter.

V. DISCUSSION

Over roughly the first 40 dB of sound levels above a typical mammal's hearing threshold, the characteristics shown in Fig. 4 are approximately the characteristics of the high-frequency cochlear filters. That the IIR filter response is linear over roughly the same input range (cf. Fig. 5) is expeditious, and to achieve "greater" linearity is not desirable in the way of achieving a fit to biological data. In the real cochlea, saturation occurs at higher sound levels, and the saturation is of a compressive nature.

For comparison purposes, Fig. 6 shows the IIR filter performance compared to its target, the TWAmp model [7], to results from a representative version of a Mead/Lyons [1] cochlea, and

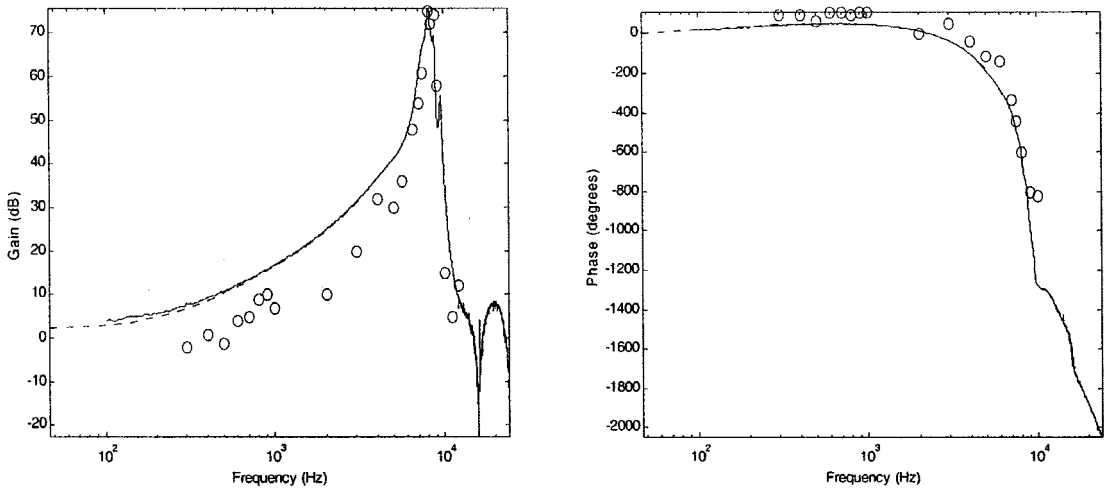

Fig. 4. Magnitude and phase response of filter in hardware (solid line), software simulation with quantized coefficients (dashed line), and biological data (circles).

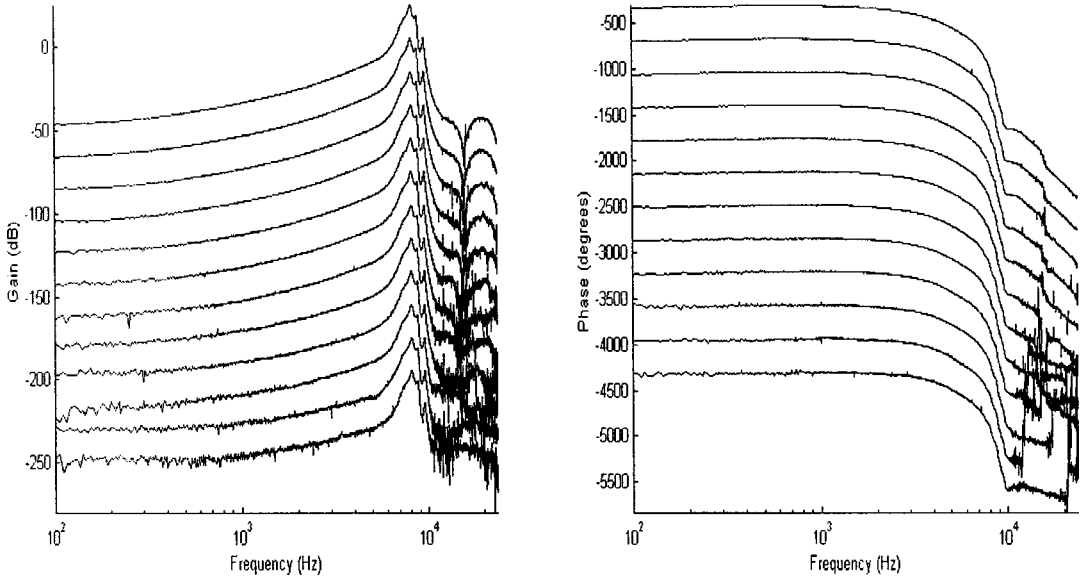

Fig. 5. Magnitude and phase response for input amplitudes of 127, 90, 64, 45, 32, 23, 16, 11, 8, 6, 4, and 3. The response curves are largely identical and in this figure they have been separated for visual clarity.

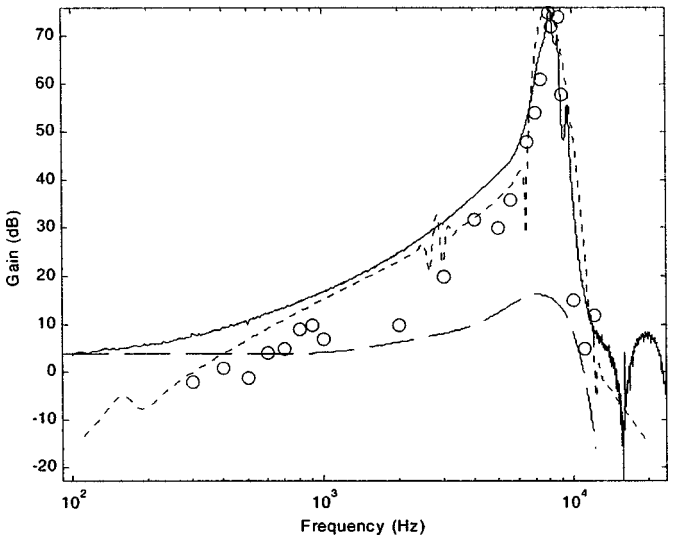

Fig. 6. Magnitude response of the IIR hardware (solid line), biological data (circles), TWAmp model (small dashes), and Mead cochlea (large dashes).

to biological data [13]. Our filter characteristic misses the biological data principally at lower frequencies. This is for two reasons. First, our IIR filter misses its target (the traveling-wave amplifier) at lower frequencies. Second, we chose the degree to which the IIR filter misses the target at low frequency, by the choice we made in deciding how many filter coefficients would be used: The low frequency side of the curve was not of great interest, and fitting it well was not considered important. The same reasoning had been applied earlier regarding the traveling-wave amplifier itself, and consequently in the same low frequency region, the traveling-wave amplifier model also misses the biological data. However, given the overwhelming difference between the IIR filter's curves in comparison with the curves from the original Lyons/Mead [1] cochlea plus the fact that results from a more recent version [2] of the Lyons/Mead cochlea are not significantly better in terms of fitting the biological data, it is easy to consider the IIR filter a considerable improvement over those designs. Moreover, compared with the traveling-wave amplifier results, we feel the IIR filter is also clearly meritorious,

because the IIR's computation is in real time and at low cost. This is compared to the approximately 0.1 h/ms computational time (DECstation 5000 at approximately 35 VAX11780 equivalents) required for the traveling-wave amplifier model [7] in the linear case.

We chose to take as the output 8 bits of a wider internal register, such that the output can never overflow given any valid, 8-bit input. We have also explored choosing lower-order bits as output, thus, achieving finer signal quantization at lower signal levels. In that case, we configured the output to saturate (hold at the maximum value) rather than overflow. In such a case, the noise at off-peak frequencies (cf. Fig. 5) is less, because internal registers of the filter are not simply setting at zero due to truncation at the low end.

Both of these modes of operation are of practical importance. First of all, our target performance was to have linear performance, and the version that cannot overflow has linear performance, albeit at the expense of increased noise at lower input signal levels. The (alternative) filter that has a saturating (clipping) output is also potentially useful for hearing applications, because in real life, the mammalian filters are nonlinear. However, they have a compressive and not necessarily a saturating characteristic, and surely if they saturate, they do not saturate so abruptly as does our digital filter.

Indeed, the worst situation is one in which the output is wrong for some unknown period of time. This egregious situation can be present in the analog VLSI (aVLSI) case: The aVLSI chips can fail when overdriven, and in our experience [2], the output may remain stuck at the rails for some much longer time period after the input signal has returned to the normal input range. If our filter were configured to allow overflow, the error would persist at the output for roughly as long as the input was out of range plus for the time it takes for the errors to clear the recursion. This is nominally forever (IIR), but from a practical standpoint the recovery time is in milliseconds in the digital case, while it can be on the order of seconds in the aVLSI case.

ACKNOWLEDGMENT

The authors would like to acknowledge R. Guida's valuable assistance.

REFERENCES

[1] C. Mead, *Analog VLSI and Neural Systems*. Reading, MA: Addison-Wesley, 1989, ch. 16.

[2] X. Xing, "Characterization and redesign of an electronic cochlea Chip," M.S. thesis, Boston Univ., Boston, MA, 2000.

[3] J. M. Kates, "A time-domain digital cochlear model," *IEEE Trans. Signal Processing*, vol. 39, no. 12, pp. 2573–2592, 1991.

[4] C. D. Summerfield and R. F. Lyon, "ASIC implementation of the lyon cochlea model," in *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Processing*, San Francisco, CA, 1992, pp. V673–V676.

[5] M. Brucke, W. Nebel, A. Schwarz, B. Mertsching, M. Hansen, and B. Kollmeier, "VLSI-implementation of a psychoacoustically and physiologically motivated speech preprocessor," in *Proc. NATO Advanced Study Institute on Computational Hearing*, Tuscany, Italy, 1998.

[6] S. C. Lim, A. R. Temple, S. Jones, and R. Meddis, "VHDL-based design of biologically inspired pitch detection system," in *Proc. IEEE Int. Conf. Neural Networks*, vol. 2, Houston, TX, 1997, pp. 922–927.

[7] A. Hubbard, "A traveling-wave amplifier model of the cochlea," *Science*, vol. 259, pp. 68–71, 1993.

[8] *Signal Processing Toolbox User's Guide (For Use With Matlab)*, The Math Works, Inc., Natick, MA, 1998.

[9] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1989.

[10] T. W. Parks and C. S. Burrus, *Digital Filter Design*. New York: Wiley, 1987, ch. 8.

[11] L. B. Jackson, *Digital Filters and Signal Processing*. Norwell, MA: Kluwer, 1996, ch. 11.

[12] J. Dattorro, "The implementation of recursive digital filters for high-fidelity audio," *J. Audio Eng. Soc.*, vol. 36, no. 11, 1988.

[13] M. A. Ruggero, N. C. Rich, L. Robles, and B. G. Shivapuja, "Middle-ear response in the chinchilla and its relationship to mechanics at the base of the cochlea," *J. Acoust. Soc. Amer.*, vol. 87, no. 4, pp. 1612–1629, 1990.

Apurva Mishra received the B.Tech. degree in electrical engineering from the Indian Institute of Technology, Bombay, India, in 1998 and the M.S. degree in computer systems engineering from Boston University, Boston, MA, in 2000.

From 1998 to 2000, he was with the Department of Electrical and Computer Engineering, Boston University. Currently, he works on microprocessor chip design within the Desktop Platforms Group at Intel Corporation. His research interests lie in VLSI design.

Allyn E. Hubbard received the B.S.E.E., M.S.E.E., and Ph.D. (E.E.) degrees from the University of Wisconsin, Madison.

He was a Postdoctoral Fellow at Johns Hopkins University, Baltimore, MD. In 1979, he joined both the Department of Electrical and Computer Systems Engineering and the Department of Biomedical Engineering, Boston University, Boston, MA. He has taught courses principally in the integrated hardware area. His research work involves the fundamental sciences with applications in chipbuilding for biotechnology applications.