EURASIP Journal on Applied Signal Processing 2003:7, 629–638
© 2003 Hindawi Publishing Corporation

# An FPGA-Based Electronic Cochlea

**M. P. Leong**
*Department of Computer Science and Engineering, The Chinese University of Hong Kong, Shatin, NT, Hong Kong*
*Email: mpleong@cse.cuhk.edu.hk*

**Craig T. Jin**
*Department of Electrical and Information Engineering, The University of Sydney, Sydney, NSW 2006, Australia*
*Email: craig@ee.usyd.edu.au*

**Philip H. W. Leong**
*Department of Computer Science and Engineering, The Chinese University of Hong Kong, Shatin, NT, Hong Kong*
*Email: phwl@cse.cuhk.edu.hk*

*Received 18 June 2002 and in revised form 1 August 2002*

A module generator which can produce an FPGA-based implementation of an electronic cochlea filter with arbitrary precision is presented. Although hardware implementations of electronic cochlea models have traditionally used analog VLSI as the implementation medium due to their small area, high speed, and low power consumption, FPGA-based implementations offer shorter design times, improved dynamic range, higher accuracy, and a simpler computer interface. The tool presented takes filter coefficients as input and produces a synthesizable VHDL description of an application-optimized design as output. Furthermore, the tool can use simulation test vectors in order to determine the appropriate scaling of the fixed-point precision parameters for each filter. The resulting model can be used as an accelerator for research in audition or as the front-end for embedded auditory signal processing systems. The application of this module generator to a real-time cochleagram display is also presented.

**Keywords and phrases:** field programmable gate array, electronic cochlea, VHDL modules.

## 1. INTRODUCTION

The field of neuromorphic engineering has the long-term objective of taking architectures from our understanding of biological systems to develop novel signal processing systems. This field of research, pioneered by Mead [1], has concentrated on using analog VLSI to model biological systems. Research in this field has led to many biologically inspired signal processing systems which have improved performance compared to traditional systems.

The human cochlea is a transducer which converts mechanical vibrations from the middle ear into neural electrical discharges, and additionally provides spatial separation of frequency information in a manner similar to that of a spectrum analyzer [2]. It serves as the front-end signal processing for all functions of the auditory nervous system such as auditory localization, pitch detection, and speech recognition.

Although it is possible to simulate cochlea models in software, hardware implementations may have orders of magnitude of improvement in performance. Hardware implementations are also attractive when the target applications are on embedded devices in which power efficiency and small-footprint are design considerations.

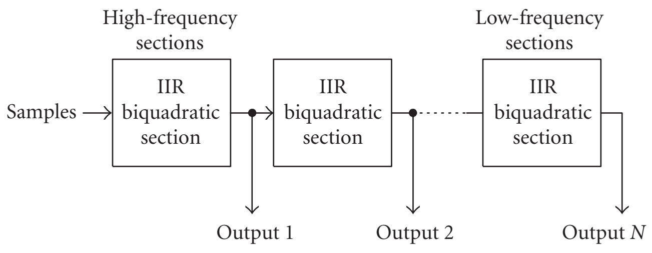


Figure 1: Cascaded IIR biquadratic section used in the Lyon and Mead cochlea model.

The electronic cochlea, first proposed by Lyon and Mead [2], is a cascade of biquadratic filter sections (as shown in Figure 1) which mimics the qualitative behavior of the human cochlea. Electronic cochlea have been successfully used in auditory signal processing systems such as spatial localization [3], pitch detection [4], a computer peripheral [5], amplitude modulation detection [6], correlation [7], and speech recognition [8].

There have been several previous implementations of electronic cochlea in analog VLSI technology. The original implementation by Lyon and Mead was published in 1988 and used continuous time subthreshold transconductance circuits to implement a cascade of 480 stages [2, 9]. In 1992, Watts et al. reported a 50-stage version with improved dynamic range, stability, matching, and compactness [10]. A problem with analog implementations is that transistor matching issues affect the stability, accuracy, and size of the filters. This issue was addressed by van Schaik et al. in 1997 using compatible lateral bipolar transistors instead of MOSFETs in parts of the circuit [11]. Their 104-stage test chip showed greatly improved characteristics. In addition, a switched capacitor cochlea filter was proposed by Bor et al. in 1996 [12].

There have also been several previously reported digital VLSI cochlea implementations. In 1992, Summerfield and Lyon reported an application-specific integrated circuit (ASIC) implementation which employed bit-serial second-order filters [13]. In 1997, Lim et al. reported a VHDL-based pitch detection system which used first-order Butterworth band pass filters for cochlea filtering [14]. Later in 1998, Brucke et al. designed a VLSI implementation of a speech preprocessor which used gammatone filter banks to mimic the cochlea [15]. The implementation by Brucke et al. used fixed-point arithmetic and they also explored trade-offs between wordlength and precision. In 2000, Watts built a 240-tap high-resolution implementation of a cochlea model using FPGA technology (http://www.lloydwatts.com/neuroscience.shtml) and in 2002 a tenth-order recursive cochlea filter was implemented using FPGA technology [16].

A field programmable gate array (FPGA) is an array of logic gates in which the connections can be configured by downloading a bitstream into its memory. Traditional ASIC design requires weeks or months for the fabrication process, whereas an FPGA can be configured in milliseconds. An additional advantage of FPGA technology is that the same devices can be reconfigured to perform different functions. At the time of writing this paper in 2002, FPGAs had equivalent densities of ten million system gates.

Since most systems which employ an electronic cochlea are experimental in nature, the long design and fabrication times associated with both analog and digital VLSI technology are a major shortcoming. Recently, FPGA technology has improved in density to the point where it is possible to develop large scale neuromorphic systems on a single FPGA. Although these are admittedly larger in area, have higher power consumption, and may have lower throughput than the more customized analog VLSI implementations, many interesting neuromorphic signal processing systems can be implemented using FPGA technology, enjoying the following advantages over analog and digital VLSI:

(i) shorter design and fabrication time;
(ii) more robust to power supply, temperature, and transistor mismatch variations than analog systems;
(iii) arbitrarily high dynamic range and signal-to-noise ratios can be achieved over analog systems;
(iv) whereas a VLSI design is usually tailored for a single application, the reconfigurability and reuseability of an FPGA enables the same system to be used for many applications;
(v) designs can be optimized for each specific instance of a problem whereas ASICs need to be more general purpose;
(vi) they can be interfaced more easily with a host computer.

The main difficulty that one faces in implementing an electronic cochlea on an FPGA is the choice of arithmetic system to be used in the imple mentation of the underlying filters. In the module generator which will be presented, a fixed-point implementation strategy was chosen over floating point since we believed it would result in an implementation with smaller area. Distributed arithmetic (DA) was used to implement the multipliers associated with the filters in an efficient manner. Finally, a module generator which can generate synthesizable VHDL descriptions of arbitrary wordlength fixed-point cochlea filters was developed. The module generator can also be used, together with our *fp* simulation tool [17, 18], to determine the minimum and maximum ranges of all variables. This range information is then used to determine the maximal number of fractional bits which can be used in the variable's two's complement fraction representation, hence minimizing quantization error.

The FPGA implementation of the electronic cochlea described here can serve as a computational accelerator in its own right, or be used as a front-end preprocessing stage for embedded auditory applications. As a sample application, a real-time cochleagram display is presented.

The rest of the paper is organized as follows. In Section 2, Lyon and Mead's cochlea model is described. Section 3 describes the implementation of the filter stages using DA. Our design methodology is presented in Section 4, followed by results in Section 5. Conclusions are drawn in Section 6.

## 2. LYON AND MEAD'S COCHLEA MODEL

Lyon and Mead proposed the first electronic cochlea in 1988 [2, 19]. This model captured the qualitative behavior of the human cochlea using a simple cascade of second order filter stages which they implemented in analog VLSI. In this section, a very superficial summary of the Lyon and Mead cochlea model is given. More detailed descriptions of the cochlea can be found in [2, 20].

The human cochlea, or inner ear, is a three-dimensional fluid dynamic system which converts mechanical vibrations from the middle ear into neural electrical discharges [2]. It is composed of the basilar membrane, inner hair cells, and outer hair cells. The cochlea connects to higher levels in the auditory pathway for further processing.

The basilar membrane is a longitudinal membrane within the cochlea. The oval window provides the input to the cochlea. Vibrations of the eardrum are coupled via bones in the middle ear to the oval window causing a traveling wave from base to apex along the basilar membrane. The basilar membrane has a filtering action and can be thought of as

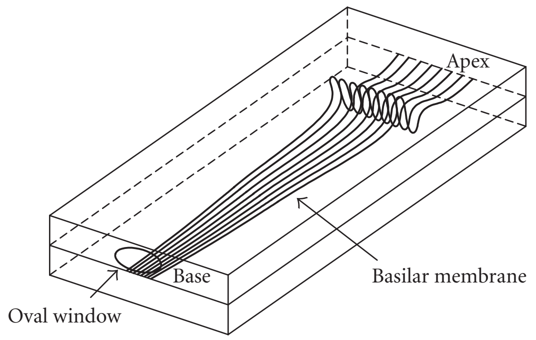


FIGURE 2: Illustration of a sine wave travelling through a simplified box model of an uncoiled cochlea (adapted from [2]).

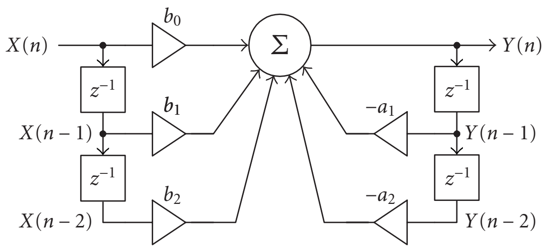


FIGURE 3: The architecture of an IIR biquadratic section.

a cascade of lowpass filters with exponentially decreasing cutoff frequency from base to apex.

The result of the filtering of the basilar membrane at any point along its length is a bandpass filtered version of the input signal, with center frequency decreasing along its length. Different distances along the basilar membrane are tuned to specific frequencies in a manner similar to that of a spectrum analyzer. A simplified box model showing a sinusoidal wave traveling along an uncoiled cochlea is shown in Figure 2.

Several thousand inner hair cells are distributed along the basilar membrane and convert the displacement of the basilar membrane to a neural signal. The hair cells also perform a half-wave rectifying function since only displacements in one direction will cause neurons to fire.

The outer hair cells perform automatic gain control by changing the damping of the basilar membrane. It is interesting to note that there are approximately three times more outer hair cells than inner hair cells.

In order to simulate the properties of the basilar membrane, Lyon and Mead's cochlea model used a cascade of scaled second-order lowpass filters with the transfer function

$$H(s) = \frac{1}{\tau^2 s^2 + (1/Q)\tau s + 1}, \tag{1}$$

where $Q$ represents the damping characteristic (or quality) of the filter and $\tau$ the time constant. In the cochlea filter, the $\tau$ of each filter is varied exponentially along the cascade, causing filters to have exponentially decreasing cutoff frequencies. The $Q$ of all the filters is held constant. The output of each filter corresponds to the displacement at different positions along the basilar membrane.

## 3. IIR FILTERS USING DA

### 3.1. Distributed arithmetic

DA offers an efficient method to implement a sum of products (SOP) provided that one of the variables does not change during execution. Instead of requiring a multiplier, DA utilizes a precomputed lookup table [21, 22].

Consider the SOP, $S$ of $N$ terms

$$S = \sum_{i=0}^{N-1} k_i x_i, \tag{2}$$

where $k_i$ is the (fixed) weighting factor and $x_i$ is the input. For two's complement fractions, the numerical value of $x_i = \{x_{i0}x_{i1} \cdots x_{i(n-1)}\}$ is

$$x_i = -x_{i0} + \sum_{b=1}^{n-1} x_{ib} \times 2^{-b}. \tag{3}$$

Substituting (3) into (2) yields

$$\begin{aligned}
S = &-(x_{00} \times k_0 + x_{10} \times k_1 + \cdots + x_{(N-1)0} \times k_{N-1}) \times 2^0 \\
&+ (x_{01} \times k_0 + x_{11} \times k_1 + \cdots + x_{(N-1)1} \times k_{N-1}) \times 2^{-1} \\
&+ (x_{02} \times k_0 + x_{12} \times k_1 + \cdots + x_{(N-1)2} \times k_{N-1}) \times 2^{-2} \\
&\quad \vdots \\
&+ (x_{0(n-1)} \times k_0 + x_{1(n-1)} \times k_1 + \cdots + x_{(N-1)(n-1)} \times k_{N-1}) \\
&\quad \times 2^{-(n-1)}.
\end{aligned} \tag{4}$$

The organization of the input variables is in a bit-serial least significant bit (LSB) first format. Since $x_{ij} \in \{0, 1\}$ ($i = 0, 1, \ldots, N-1$, $j = 0, 1, \ldots, n-1$), each term within the brackets of (4) is the sum of weighting factors $k_0, k_1, \ldots, k_{N-1}$. On every clock cycle, one of the bracketed terms of $S$ can thus be computed by applying $x_0, x_1, \ldots, x_{N-1}$ as the address inputs of a $2^{(N-1)}$ entry read-only memory (ROM). The contents of the ROM are precomputed from the constant $k_i$'s and are shown in Table 1. The output of the ROM is multiplied by a power of two (a shift operation) and then accumulated. After $n$ cycles, the accumulator contains the value of $S$.

### 3.2. Digital IIR filters

A general IIR second-order filter has a transfer function of the form

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}}. \tag{5}$$

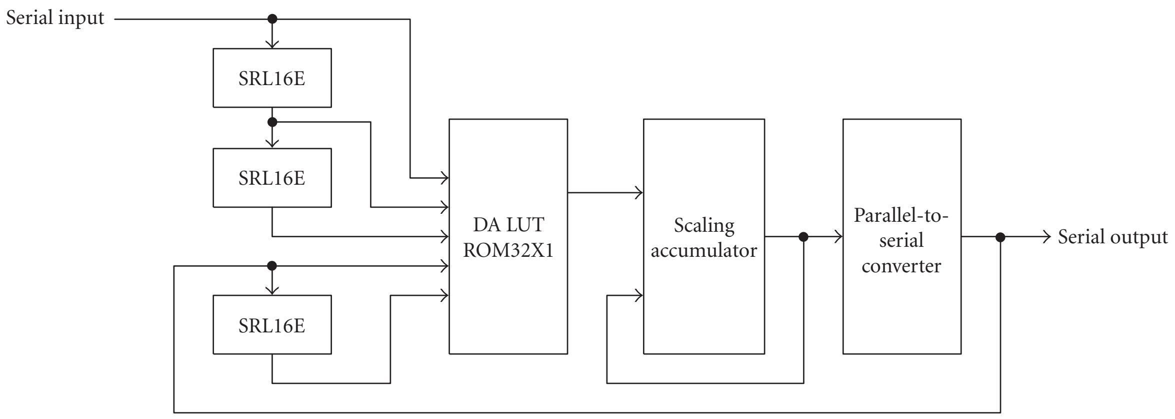


FIGURE 4: Implementation of an IIR biquadratic section on an Xilinx Virtex FPGA.

TABLE 1: Contents of a DA ROM. For each address, the terms $k_i$ for which $b_i = 1$ are summed.

| $b_{N-1} \cdots b_2 b_1 b_0$ | Address | Contents |
|---|---|---|
| $0 \cdots 000$ | 0 | 0 |
| $0 \cdots 001$ | 1 | $k_0$ |
| $0 \cdots 010$ | 2 | $k_1$ |
| $0 \cdots 011$ | 3 | $k_0 + k_1$ |
| $0 \cdots 100$ | 4 | $k_2$ |
| $0 \cdots 101$ | 5 | $k_0 + k_2$ |
| $0 \cdots 110$ | 6 | $k_2 + k_1$ |
| $0 \cdots 111$ | 7 | $k_0 + k_1 + k_2$ |
| $\vdots$ | $\vdots$ | $\vdots$ |
| $1 \cdots 111$ | $2^{N-1}$ | $k_0 + k_1 + \cdots + k_{N-1}$ |

The corresponding time domain IIR filter can be implemented by the function

$$y(n) = b_0 x(n) + b_1 x(n-1) + b_2 x(n-2) + a_0 y(n-1) + a_1 y(n-2), \tag{6}$$

where $x(n-k)$ is the $k$th previous input, $y(n-k)$ is the $k$th previous output, and $y(n)$ is the output. The operation is essentially the SOP of five terms, and can be directly mapped to a biquadratic section as shown in Figure 3.

Figure 4 illustrates our actual implementation using DA (described in Section 3.1) on an Xilinx Virtex FPGA. The previous values $x(n-1)$, $x(n-2)$, and $y(n-2)$ are implemented using shift registers with the number of stages equal to the wordlength of the variables used. The shift registers are implemented by cascades of Virtex SRL16E primitives for minimum area. The DA ROM takes $x(n)$, $x(n-1)$, $x(n-2)$, $y(n-1)$, and $y(n-2)$ as inputs to generate partial sums (bracketed terms in (4)). As there are 5 inputs, the required number of entries in the ROM is $2^5 = 32$, leading to an efficient implementation using Xilinx ROM32X1 primitives. The scaling accumulator shifts and adds the output from the ROM (unscaled partial sum in bit-parallel organization) at every cycle to produce $y(n)$. In the last cycle of scaling and accumulation, the parallel-to-serial converter latches the value at the scaling accumulator. Since the scaling accumulator has a latency equal to the wordlength of the variables, the value latched by the converter is $y(n-1)$.

## 4. DESIGN METHODOLOGY

Given the filter coefficients, the designer selects appropriate values of filter wordlength and the number of bits (width) of the DA ROM's output. Note that all filter sections have the same wordlength although the allocation of integer and fractional parts used within each filter section can vary.

The cochlea filter model is written in a subset of C which supports only expressions and assignments [17, 18]. A compiler uses standard parsing techniques to translate expressions into directed acyclic graphs (DAG). Each operator is mapped to a module which is a software object consisting of a set of parameters, a simulator, and a component generator. The simulator can perform the operation at a requested precision to determine range information. It can also compare fixed-point output with a floating-point computation to derive error statistics.

As an input, the *fp* cochlea generator takes the coefficients obtained from an auditory toolbox, the wordlength of variables, and the width of the DA ROM. Although inputs and outputs of all filter sections are of the same wordlength, their fractional wordlength can be different (two's complement fractions are used). The dynamic ranges of inputs and outputs are determined by *fp* through simulation of a set of user-supplied test vectors. The generator performs simulation using the test vectors as inputs and the range of each variable can be determined. From this information, the minimum number of bits needed for the integer part of each variable is known and, since the wordlength is fixed, the maximum number of bits can be assigned to the fractional part of the variable.

After deducing the best representation for each variable, the generator outputs a synthesizable VHDL code that describes an implementation of the corresponding cochlea model. The fractional wordlengths of the scaling accumulator and the output variable can be different, so the operator

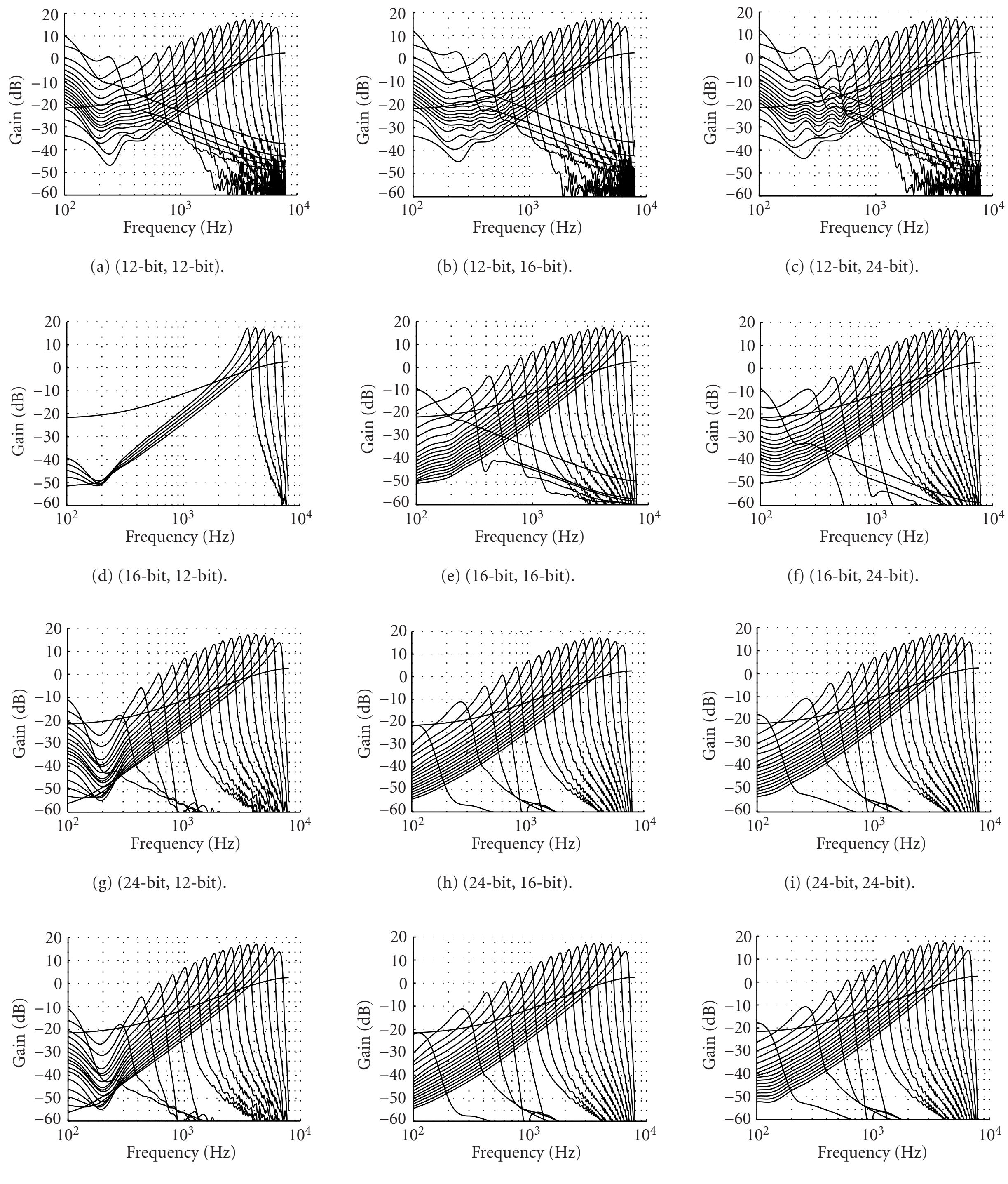
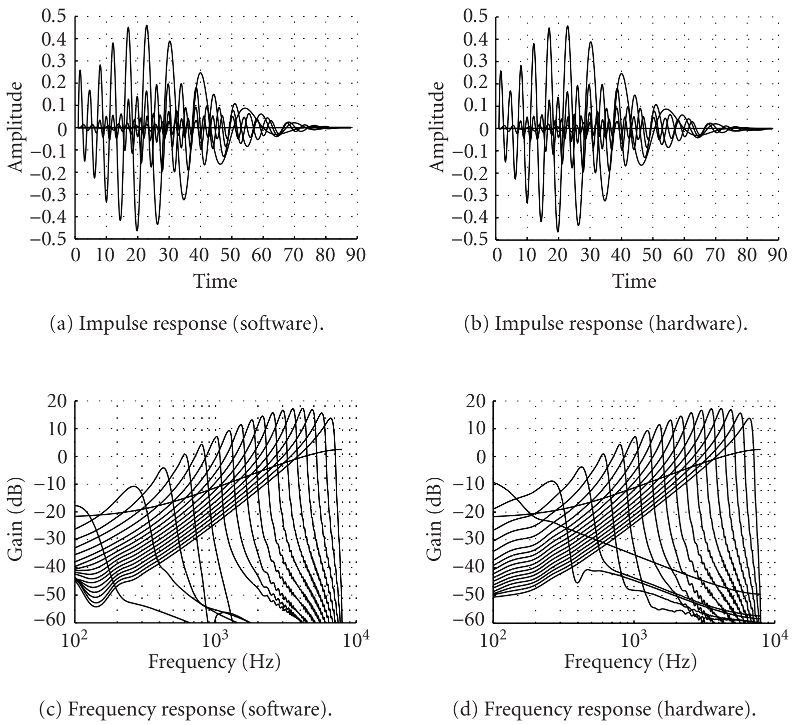


Figure 5: Frequency responses of cochlea implementations with different wordlength and width of ROMs (wordlength, ROM width).

must also include a mechanism to convert the former to the latter. Since the output of the scaling accumulator is bit-parallel while the output variable is bit-serial, the parallel-to-serial converter can perform format scaling by selecting the appropriate bits to serialize. The resulting VHDL description can then be used as a core in other designs.

The high level cochlea model description is approximately 60 lines of C code. From that, it generates approximately 50000 lines of VHDL code for the case of a cochlea filter with 88 biquadratic sections.



(a) Impulse response (software).
(b) Impulse response (hardware).
(c) Frequency response (software).
(d) Frequency response (hardware).

Figure 6: Impulse response of (a) the software floating-point implementation and (b) hardware 16-bit wordlength, 16-bit ROM width implementation. Frequency response of (c) the software floating-point implementation and (d) hardware 16-bit wordlength, 16-bit ROM width implementation.

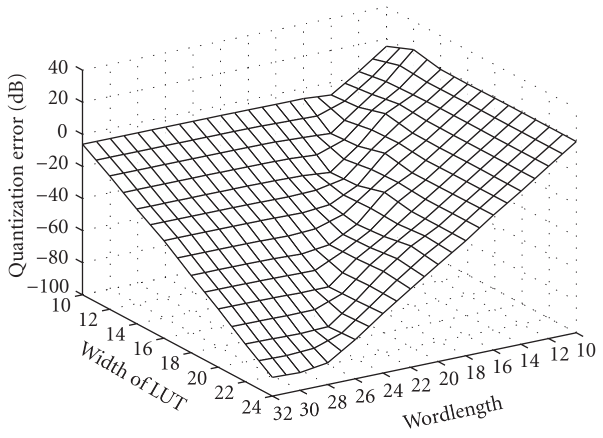


Figure 7: Mesh plot showing the quantization errors of implementations with varying wordlengths and DA ROM widths.

## 5. RESULTS

The cochlea implementation was tested on an Annapolis "Wildstar" Reconfigurable Computing Engine [23] which is a PCI-based reconfigurable computing platform containing three Xilinx Virtex XCV1000-BG560-6 FPGAs. The cochlea implementations were verified by comparing Synopsys VHDL Simulator simulations with the results produced by a floating-point software model. Synthesis and implementation were performed using Synopsys FPGA Express 3.4 and Xilinx Foundation 3.2i, respectively.

### 5.1. *Trade-offs among wordlength, width of DA ROM, and precision*

The coefficients for the biquadratic filters in our implementation of Lyon and Mead's cochlea model were obtained using Slaney's Auditory Toolbox [24]. This Matlab toolbox has several different cochlea models, test inputs, and visualization tools. The same toolbox was used to verify our designs and produce cochleagram plots.

The coefficients of these implementations were obtained from the Auditory Toolbox using the Matlab command `DesignLyonFilters(16000, 8, 0.25)`, which specifies a 16 kHz sampling rate, $Q = 8$, and a spacing which gives 88 biquadratic filters. A series of cochlea implementations, with wordlengths from 10 to 32 bits and DA ROM width from 10 to 24 bits, was generated in order to present the trade-offs among wordlengths, widths of DA ROMs, and precisions.

In order to present the improvement in precision with increasing wordlengths and ROM width, the frequency responses of several different fixed-point implementations are plotted in Figure 5. Figure 6 shows impulse and frequency responses obtained from a software floating-point implementation, a hardware 16-bit wordlength, and 16-bit ROM width implementation.

It can be observed that the filter accuracy gradually improves with increasing wordlength or ROM width. When

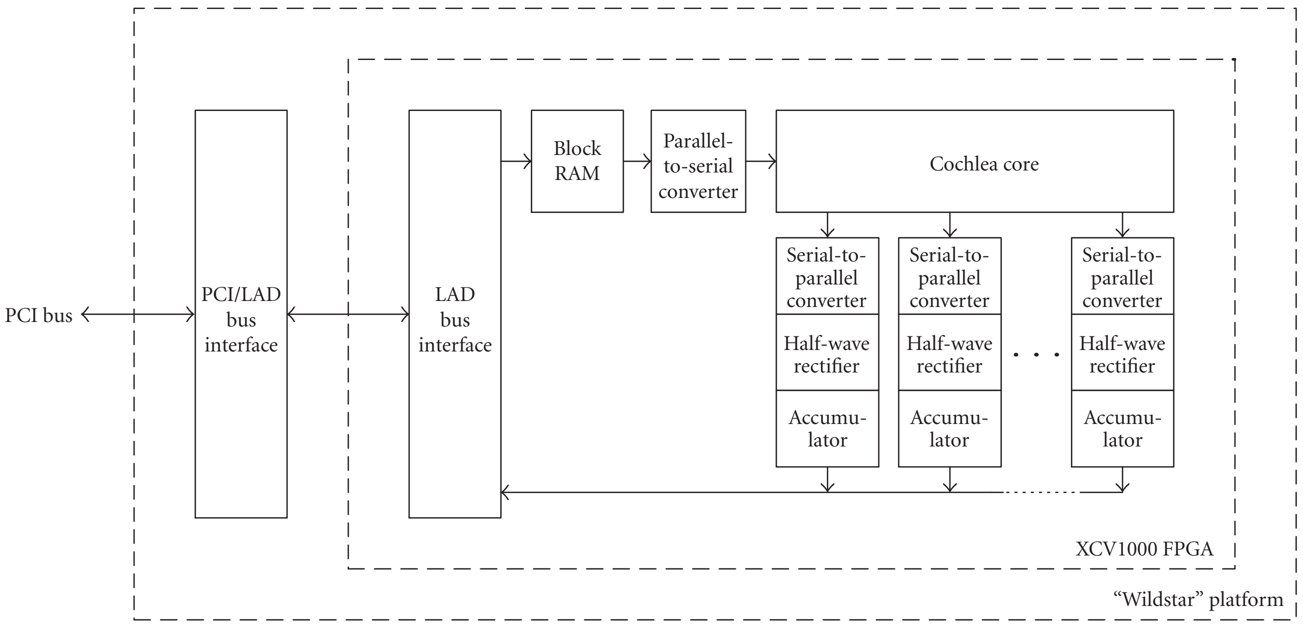


FIGURE 8: System architecture of the cochleagram display.

TABLE 2: Area requirements of an 88-section cochlea implementation of different wordlengths and ROM width (number of slices).

| Wordlength | ROM Width | | | |
|---|---|---|---|---|
| | 12-bit | 16-bit | 20-bit | 24-bit |
| 12-bit | 5770 | 6582 | 7440 | 8340 |
| 16-bit | 6160 | 6800 | 7589 | 8515 |
| 20-bit | 6914 | 7343 | 7874 | 8602 |
| 24-bit | 7620 | 8048 | 8578 | 9106 |
| 28-bit | 8288 | 8748 | 9278 | 9805 |
| 32-bit | 9297 | 9716 | 10245 | 10771 |

TABLE 3: Maximum clock rates and corresponding sampling rates of 88-section cochlea implementations of different wordlengths and ROM width (maximum clock rate (MHz) and maximum sampling rate (MHz)).

| Wordlength | ROM Width | | | |
|---|---|---|---|---|
| | 12-bit | 16-bit | 20-bit | 24-bit |
| 12-bit | 56.42, 4.70 | 62.79, 5.23 | 69.49, 5.79 | 67.24, 5.60 |
| 16-bit | 67.48, 4.22 | 67.54, 4.22 | 65.16, 4.07 | 65.02, 4.06 |
| 20-bit | 64.48, 3.22 | 63.58, 3.18 | 61.86, 3.09 | 61.79, 3.09 |
| 24-bit | 64.24, 2.68 | 60.98, 2.54 | 57.94, 2.41 | 59.47, 2.48 |
| 28-bit | 60.22, 2.15 | 57.93, 2.07 | 54.00, 1.93 | 49.09, 1.75 |
| 32-bit | 62.68, 1.96 | 63.11, 1.97 | 65.23, 2.04 | 63.09, 1.97 |

wordlengths or ROM widths are too small, there are significant quantization effects that may result in oscillation (as in the 12-bit wordlength implementations) or improper frequency responses at certain frequency intervals (as in the 12-bit DA ROM implementations). With 24-bit wordlength and 16-bit ROMs, for example, the total quantization error is −39.46 dB, which is sufficient for most speech applications. Figure 7 shows the trend of improved quantization error with increasing wordlength and ROM width.

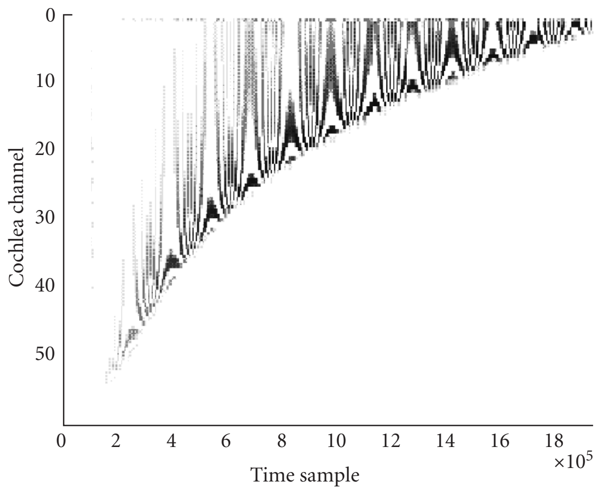
Area requirements, maximum clock rates, and maximum sampling rates of these implementations on a Xilinx Virtex XCV1000-6 FPGA, as reported by the Xilinx implementation tools, are shown in Tables 2 and 3. A Xilinx XCV1000 has 12288 slices and the largest currently available parts, XCV3200, have 32448 slices. As a bit-serial architecture was employed, the effective sampling rates of the implementations are their maximum clock rates divided by their wordlengths.

### 5.2. *Application to a cochleagram display*

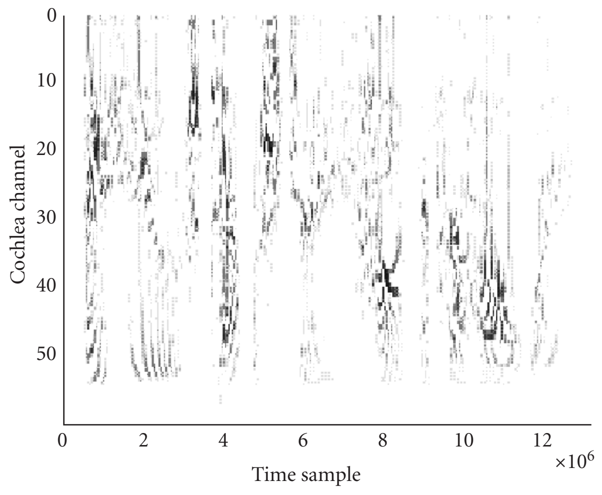
A 24-bit wordlength, 16-bit DA ROM implementation was used to construct a cochleagram display application. Due to limited hardware resources on a Xilinx XCV1000-6 FPGA, only the first 60 out of the 88 cochlea sections were used. These cochlea sections correspond to a frequency range of 1006 to 7630 Hz.

The design of the cochleagram display is shown in Figure 8. The host PC writes input data into a dual-port block RAM (256 × 32-bit synchronous RAM) which passes through a parallel-to-serial converter and enters the cochlea core. Each of the outputs of the cochlea core undergoes serial-to-parallel conversion followed by half-wave rectification (to model the functionality of the inner hair cells). The outputs were integrated over 256 samples and sent to the PC for display.

The cochleagram display was tested with several different inputs. Figure 9 shows the cochleagrams produced from swept-sine wave and the auditory toolbox's "tapestry" inputs; the former is a 25-second linear chirp and the latter is the speech file of a woman saying "a huge tapestry hung in her hallway."



(a) Swept-sine wave.



(b) “tapestry” input.

Figure 9: Cochleagrams of (a) swept-sine wave and (b) “tapestry” inputs. The former has 400000 samples while the latter has 50380 samples. Only the first 60 out of the 88 cochlea sections were used because of limited hardware resources on a Xilinx XCV1000-6 FPGA. These cochlea sections correspond to a frequency range of 1006–7630 Hz.

The cochleagram display requires 10344 slices and can be clocked at 44.15 MHz, yielding a sampling rate of 1.84 MHz (or 115 times faster than real-time performance). Including software and interfacing overheads, the measured throughput on the “Wildstar” platform was 238 kHz. As a comparison, the auditory toolbox achieves a 64 kHz throughput on a Sun Ultra-5 360 MHz machine. The performance could be further improved by using large and/or faster speed grade FPGAs, or via improved floorplanning of the design which would allow a higher clock frequency.

It is interesting to compare the FPGA-based cochleagram system with a similar system developed in analog VLSI by Lazzaro et al. in 1994 [5]. Using a 2 $\mu$m CMOS process, they integrated a 119 stage silicon cochlea (with a slightly more sophisticated hair cell model), nonvolatile analog storage, and a sophisticated event-based communications protocol on a single $3.6 \times 6.8\,\text{mm}^2$ chip with a power consumption of 5 mW. The analog VLSI version has improved density and power consumption compared with the FPGA approach. However, the FPGA version is vastly simpler, easier to modify, has a shorter design time, and is much more tolerant of supply voltage, temperature, and transistor matching variations. Although qualitative results are not available, it is expected that the FPGA version also has better filter accuracy, has a wider dynamic range and can operate at higher $Q$ without instability.

We believe that there are many applications of the FPGA cochlea including the development of more refined cochlea or cochlea-like models. An FPGA cochlea is particularly suited as a testbed for algorithms that involve concurrent processing across cochlea channels such as (i) more realistic hair cell models, (ii) auditory streaming and the separation of foreground stimuli from background noise, (iii) auditory processing in reverberant environments, (iv) human sound localization, and (v) bat echolocation. In addition, the FPGA platform provides an avenue for developing, simulating, and studying auditory processing in more complicated, but realistic acoustic environments, that involve multiple sound sources, multiple reflection paths, and external ear acoustic filtering that varies with sound direction. The signal processing required to simulate such realistic environments is computationally intensive and some of this preprocessing can be incorporated into the FPGA platform enabling real-time studies of auditory processing under realistic acoustic conditions. We are also interested in finding ways in which FPGA cochlea models can assist in adapting or translating cochlea processing principles into engineering implementations. Future projects include using the FPGA cochlea in comparisons of a cochlea model with an analysis-synthesis filter bank as used in perceptual audio coding, audio visualization displays, and a neuromorphic isolated word spotting system with cochlea preprocessing. Although modern digital signal processors (DSP) are capable of achieving similar or even higher performance, the FPGA may have advantages in terms of power consumption and smaller footprint. The availability of more than 100 dedicated high-speed multipliers in newer Virtex II [25] devices would enable implementations with much higher throughput than the implementation presented in this paper and would also free up more FPGA logic resources for implementing hair cell and higher-level processing models.

## 6. CONCLUSION

A parameterized FPGA implementation of an electronic cochlea that can be used as a building block for many systems which model the human auditory pathway was developed. This electronic cochlea demonstrates the feasibility of incorporating large neuromorphic systems on FPGA devices. Neuromorphic systems employ parallel distributed processing which is well suited to FPGA implementation, and may offer significant advantages over conventional architectures.

FPGAs provide a very flexible platform for the development of experimental neuromorphic circuits and offer advantages in terms of faster design time, faster fabrication time, wider dynamic range, better stability, and simpler computer interface over analog VLSI implementations.

## ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewer. The work described in this paper was supported by a direct grant from the Chinese University of Hong Kong (Project code 2050240), the German Academic Exchange Service, and the Research Grants Council of Hong Kong Joint Research Scheme (Project no. G_HK010/00).

## REFERENCES

[1] C. Mead, *Analog VLSI and Neural Systems*, Addison-Wesley, Boston, Mass, USA, 1989.

[2] R. F. Lyon and C. Mead, "An analog electronic cochlea," *IEEE Trans. Acoustics, Speech, and Signal Processing*, vol. 36, no. 7, pp. 1119–1134, 1988.

[3] J. P. Lazzaro and C. Mead, "Silicon models of auditory localization," *Neural Computation*, vol. 1, pp. 47–57, Spring 1989.

[4] J. P. Lazzaro and C. Mead, "Silicon models of pitch perception," *Proc. National Academy of Sciences*, vol. 86, no. 23, pp. 9597–9601, 1989.

[5] J. P. Lazzaro, J. Wawrzynek, and A. Kramer, "Systems technologies for silicon auditory models," *IEEE Micro*, vol. 14, no. 3, pp. 7–15, 1994.

[6] A. van Schaik and R. Meddis, "Analog very large-scale integrated (VLSI) implementation of a model of amplitude-modulation sensitivity in the auditory brainstem," *Journal of the Acoustical Society of America*, vol. 105, no. 2, pp. 811–821, 1999.

[7] C. A. Mead, X. Arreguit, and J. P. Lazzaro, "Analog VLSI model of binaural hearing," *IEEE Transactions on Neural Networks*, vol. 2, no. 2, pp. 230–236, 1991.

[8] J. P. Lazzaro, J. Wawrzynek, and R. P. Lippmann, "Micro power analog circuit implementation of hidden Markov model state decoding," *IEEE Journal Solid State Circuits*, vol. 32, no. 8, pp. 1200–1209, 1997.

[9] R. F. Lyon, "Analog implementations of auditory models," in *Proc. DARPA Workshop on Speech and Natural Language*, Morgan Kaufman, Pacific Grove, Calif, USA, February 1991.

[10] L. Watts, D. A. Kerns, R. F. Lyon, and C. A. Mead, "Improved implementation of the silicon cochlea," *IEEE Journal Solid State Circuits*, vol. 27, no. 5, pp. 692–700, 1992.

[11] A. van Schaik, E. Fragnière, and E. Vittoz, "Improved silicon cochlea using compatible lateral bipolar transistors," in *Advances in Neural Information Processing Systems*, vol. 8, MIT press, Cambridge, Mass, USA, 1997.

[12] J.-C. Bor and C.-Y. Wu, "Analog electronic cochlea design using multiplexing switched-capacitor circuits," *IEEE Transactions on Neural Networks*, vol. 7, no. 1, pp. 155–166, 1996.

[13] C. D. Summerfield and R. F. Lyon, "ASIC implementation of the Lyon cochlea model," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing*, pp. 673–676, San Francisco, Calif, USA, March 1992.

[14] S. C. Lim, A. R. Temple, and S. Jones, "VHDL-based design of biologically inspired pitch detection system," in *Proc. IEEE International Conference on Neural Network*, vol. 2, pp. 922–927, Houston, Tex, USA, June 1997.

[15] M. Brucke, W. Nebel, A. Schwarz, B. Mertsching, M. Hansen, and B. Kollmeier, "Digital VLSI-implementation of a psychoacoustically and physiologically motivated speech preprocessor," in *Proc. NATO Advanced Study Institute on Computational Hearing*, pp. 157–162, Il Ciocco, Italy, 1998.

[16] A. Mishra and A. E. Hubbard, "A cochlear filter implemented with a field-programmable gate array," *IEEE Trans. on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 49, no. 1, pp. 54–60, 2002.

[17] M. P. Leong, M. Y. Yeung, C. K. Yeung, C. W. Fu, P. A. Heng, and P. H. W. Leong, "Automatic floating to fixed point translation and its application to post-rendering 3D warping," in *Proc. 7th IEEE Symposium on Field-Programmable Custom Computing Machines*, pp. 240–248, Napa, Calif, USA, April 1999.

[18] M. P. Leong and P. H. W. Leong, "A variable-radix digit-serial design methodology and its applications to the discrete cosine transform," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 11, no. 1, pp. 90–104, 2003.

[19] R. F. Lyon and C. Mead, "Electronic cochlea," in *Analog VLSI and Neural Systems*, Addison-Wesley, Boston, Mass, USA, 1989, Chapter 16.

[20] J. O. Pickles, *An Introduction to the Physiology of Hearing*, Academic Press, London, UK, 1988.

[21] G. R. Goslin, *A Guide to Using Field Programmable Gate Arrays (FPGAs) for Application-Specific Digital Signal Processing Performance*, Xilinx, San Jose, Calif, USA, 1995, Application Note.

[22] Xilinx, *The Role of Distributed Arithmetic in FPGA-based Signal Processing*, November 1996, http://www.xilinx.com.

[23] Annapolis Micro Systems, *Wildstar Reference Manual*, 2000, Revision 3.3.

[24] M. Slaney, "Auditory toolbox: A MATLAB Toolbox for auditory modeling work," Tech. Rep. 1998-010, Interval Research Corporation, Palo Alto, Calif, USA, 1998, Version 2.

[25] Xilinx, *Virtex-II Platform FPGA User Guide*, 2001, Version 1.1.

**M. P. Leong** received his B.E. and Ph.D. degrees from The Chinese University of Hong Kong in 1998 and 2001, respectively. He is currently the System Manager of the Center for Large-Scale Computation at the same University. His research interests include network security, parallel computing, and field-programmable systems.

**Craig T. Jin** received his B.S. degree (1989) from Stanford University, M.S. degree (1991) from Caltech, and Ph.D. degree (2001) from the University of Sydney. He is currently a Lecturer at the School of Electrical and Information Engineering at the University of Sydney. Together with André van Schaik, he heads the Computing and Augmented Reality Laboratory. He is the author of over thirty technical papers and three patents. His research interests include multimedia signal processing, 3D audio engineering, programmable analogue VLSI filters, and reconfigurable computing.

**Philip H. W. Leong** received the B.S., B.E., and Ph.D. degrees from the University of Sydney in 1986, 1988, and 1993, respectively. In 1989, he was a Research Engineer at AWA Research Laboratory, Sydney, Australia. From 1990 to 1993, he was a postgraduate student and Research Assistant at the University of Sydney, where he worked on low-power analog VLSI circuits for arrhythmia classification. In 1993, he was a Consultant of SGS Thomson Microelectronics in Milan, Italy. He was a Lecturer at the Department of Electrical Engineering, University of Sydney from 1994 to 1996. He is currently an Associate Professor at the Department of Computer Science and Engineering at the Chinese University of Hong Kong and the Director of the Custom Computing Laboratory. He is the author of more than fifty technical papers and three patents. His research interests include reconfigurable computing, digital systems, parallel computing, cryptography, and signal processing.