A Stereo Asynchronous Digital Sample-Rate Converter for Digital Audio

Robert Adams and Tom Kwan

Abstract— **The design of an asynchronous digital sample-rate converter for digital-audio applications is presented. The theory of asynchronous sample-rate conversion is discussed using a signal-processing model that is based on highly interpolated input samples. A novel closed-loop address-tracking system is disclosed that solves the problem of clock-edge arrival estimation while at the same time providing a low-jitter selection of the correct polyphase filter for each output sample. Sample-rate ratio changes of up to 2:1 in either direction can be accommodated. The proposed signal-processing algorithm has been implemented in a 0.8-μm CMOS technology. Measurement results show excellent agreement with theory.**

I. Introduction

As the audio world moves more and more towards completely digital systems, the problem of interfacing equipment with differing sampling rates has grown to severe proportions. In the past, there was usually only a single digital audio processor in any particular signal chain, and the degradation caused by using the internal A/D and D/A converters of the processor was not considered excessive. However, now it is common to find all-digital studios where the signal is digitized immediately after the microphone (or other original analog source) and audio recording, production, editing and processing stays completely in the digital domain. For this reason, the need for simple digital interfacing between different digital audio equipment has become acute.

The most common solution to the digital-interface problem is to use a phase-locked loop to recover the serial bit clock of the serial data input, and to use this high-frequency clock as the internal system clock. One problem that often arises is that the system clock frequency is fixed and is not related to the incoming serial-data frequency. For example, in digital video tape recorders the internal clock must be related to some standard video frequency and must also lock to the frequency of a master video sync generator whose output is not related to the digital-audio bit-stream frequency. Therefore, such equipment must include a sample-rate converter to convert the external digital-audio signal to the local fixed sampling rate.

There are two classes of sample-rate converters; synchronous and asynchronous. In synchronous sample rate converters, the incoming sample rate needs to be converted to a new sample rate by a ratio of simple integers (3:2, for example). While such a device is useful in many applications, it suffers from the problem that its digital output is still the system clock "master" which means that output samples are produced at a fixed rate related to the input rate, and equipment which uses this data must still lock to it. Asynchronous sample rate converters, on the other hand, are designed to receive incoming data and produce output data when requested by the system. It is capable of converting between any two sample rates, and the ratio of these rates may be irrational. Further, the input and output sample rates may vary slowly over time, and the output data will correctly follow these variations.

Numerous topologies for synchronous and asynchronous sample-rate conversion have appeared in the literature [1]–[4]. They usually require multiple DSP chips as well as external phase-locked loops and are therefore not well suited for single-chip VLSI implementation.

The problem of asynchronous sample-rate conversion requires getting an accurate time-of-arrival measurement of the user-supplied clock signals. This was addressed by McNally *et al.* [1], who proposed a method of averaging instantaneous measurements over time to improve the measurement accuracy. This technique allows a lower frequency measurement clock to be used, and depending on the time-constant of the averaging, jitter on the input clocks is largely rejected. Most of the methods found in the literature decouple the clock measurement problem from the problem of memory management. This can cause errors in the form of underflowing or overflowing the memory buffer given large measurement errors or step changes in input or output sample rates. The method proposed in this paper combines both time-averaging and closed-loop memory buffer flow control in a single structure to allow error-free tracking of large variations in sample rates. In addition, there is no requirement for a high-frequency clock that is synchronized to either the input or output rate, eliminating the need for analog phase-locked loops. In the following, the theory of sample-rate conversion is discussed in Section II using a highly interpolated signal-processing model. Section III gives a practical implementation of this sample-rate conversion technique while measured results are presented in Section IV. Finally, conclusions are drawn in Section V.

II. Sample-Rate Conversion Fundamentals

The simplest way to change from one sample rate to another is to use a D/A converter in combination with a a "brick wall" filter (which ideally removes all signal images) to obtain a signal in the analog domain. The analog signal is then converted back to a digital format using an A/D converter which runs at the output sample rate. This is shown in Fig. 1(a). All-digital sample-rate converters follow the same

Manuscript received August 20, 1993; revised November 11, 1993.
R. Adams is with Analog Devices, Wilmington, MA 01887.
T. Kwan is with Analog Devices, Santa Clara, CA 95052.
IEEE Log Number 9215393.

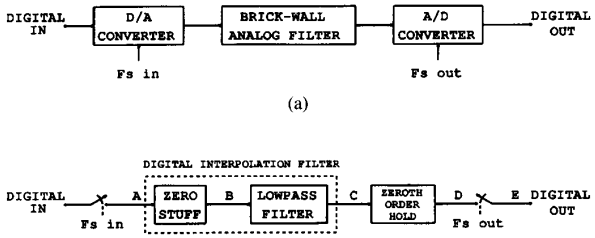

Fig. 1. (a) Analog method of sample-rate conversion; (b) an all-digital method of sample-rate conversion.

general principle but implement all the operations digitally. The analog filter may be replaced by a combination of a digital interpolation filter with a very high output sample-rate and a zero-order-hold (ZOH) to convert the discrete-time output of the digital filter to a continuous-time signal. A block diagram of an all-digital sample-rate converter (SRC) is shown in Fig. 1(b).

A. Time-Domain Analysis

Fig. 2 presents a time-domain view of various signals in an all-digital SRC as shown in Fig. 1(b). The digital interpolation filter produces a stream of output samples (trace C) that are on a much finer time grid than the original input samples (trace A). When these interpolated values are fed into a ZOH and then asynchronously resampled (at the vertical dotted bars in trace D) by the output switch, the output values represent the "nearest" (in time) values produced by the interpolation filter. There is always some error in the output samples due to the fact that the output sampling switch does not close at a time that exactly corresponds to a point on the fine time grid of the interpolated outputs.

This error can be made arbitrarily small by using a very large interpolation ratio. As the interpolation ratio is increased. the difference between adjacent interpolated samples become very small. If some output error criteria is chosen, the interpolation ratio can be chosen such that adjacent values are guaranteed to be within this range. For example, to make a sample-rate converter with 16-b accuracy, the difference between any two adjacent interpolated values should be less than 1 LSB at the 16-b level. The required interpolation ratio to achieve 16-b accuracy can be found by assuming an input signal that will cause the worst-case sample-to-sample difference in the interpolated output. Let the maximum output levels be ± 1 V for a worst case 20 kHz sine wave. The peak slew rate is $2\pi \times 20$ kHz or 125,600 V/sec. The required time grid resolution (T_{interp}) to achieve an error of less than $2/2^{16}$, given the slew-rate above, is 240 ps. Assuming an input sample rate of about 50 kHz, this gives an interpolation ratio of 83,333. This number is conservative for two reasons. One is that the peak error is used versus the root-mean-squared (RMS) error over the entire sine-wave. Also, this analysis assumes that the misalignment error, introduced by the output sampler with respect to the interpolation time grid, is maximum at each output sampling time, when it should be uniformly

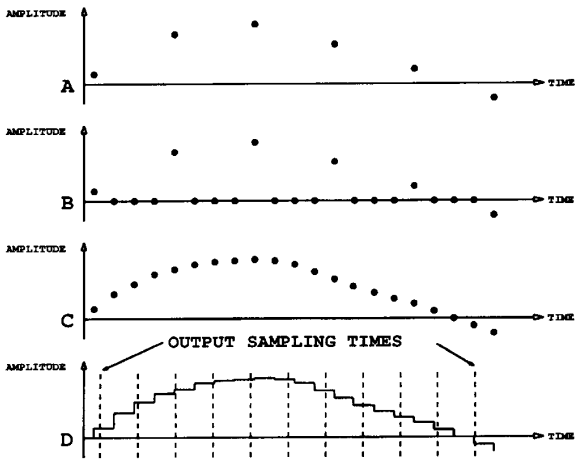

Fig. 2. Time-domain view of sample-rate conversion.

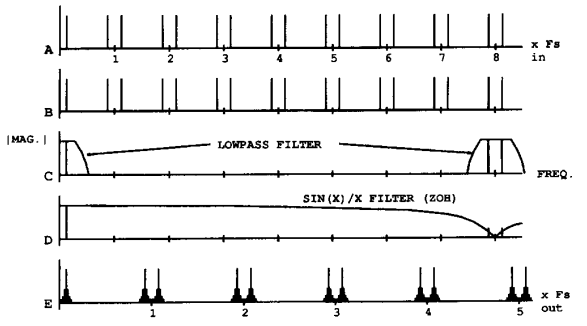

Fig. 3. Frequency-domain view of sample-rate conversion.

distributed which will lower the RMS error. For these reasons an interpolation value of 2^{16} (65536) is chosen to achieve 16-b accuracy, which gives an RMS error of 0.43 LSB's. Accuracy at lower frequencies and/or levels is significantly higher due to the $\sin(x)/x$ nature of the ZOH frequency response.

B. Frequency-Domain Analysis

The required interpolation ratio to achieve a given accuracy can also be found using frequency domain analysis. Fig. 3 shows the spectra at the output of the first sampler (A), the zero-stuff block (B), the lowpass filter (C), the ZOH (D), and the second sampler (E) given a full scale sine wave input. The interpolation filter removes all images except those about the new sample rate (Fs_{interp}) given by $N \times Fs_{\text{in}}$, where N is the interpolation ratio and Fs_{in} is the input rate (see trace C). The ZOH reduces the remaining images of the digital filter output (only one set shown in trace D). Thus, the attenuation of the ZOH at the worst case input frequency of 20 kHz determines the final quality of the output signal, as these image components are folded down upon resampling by the output switch (trace E). Given N, the worst case ZOH attenuation occurs at $N \times Fs_{\text{in}} \pm$ 20 kHz. The frequency response of a ZOH is given by $\sin(\pi f/Fs_{\text{interp}})/(\pi f/Fs_{\text{interp}})$. Using N equal 2^{16}, a sampling frequency of $2^{16} \times 50$ kHz and $f = 2^{16} \times$

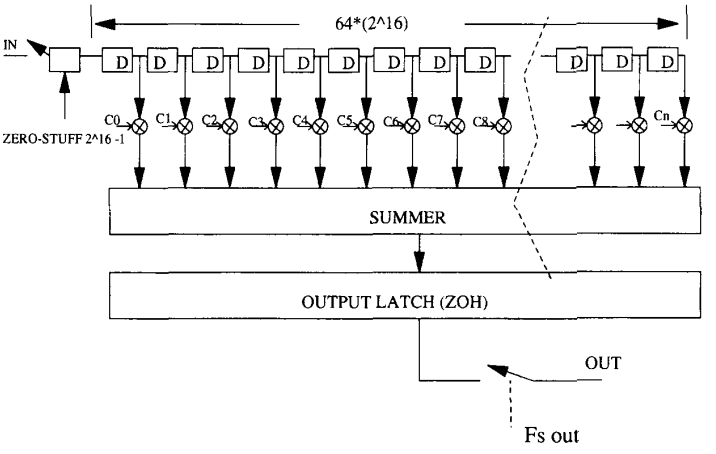

Fig. 4. Conceptual hardware model.

50 kHz ± 20 kHz, we get an image attenuation at $2^{16}\times$ 50 kHz ± 20 kHz of −103.2 and −105.6 dB, respectively. A similar analysis for the second image regions at $2Fs_{\text{interp}}$ ± 20 kHz gives −108 and −113 dB, respectively. The RMS sum of the first two image regions is −100.1 dB. Other image regions at $3Fs_{\text{interp}}$ ± 20 kHz and higher become insignificant. Note that if the output resampling rate is not equal to the input sampling rate, the folded-down images will all appear at different frequencies, justifying the use of an RMS sum in the previous calculation. This confirms the results of the time-domain analysis section that an interpolation ratio of 2^{16} is adequate for 16-b performance with the worst-case input signal (full-level 20 kHz input).

III. VLSI Implementation

A. Conceptual Hardware Model

Fig. 4 shows an obviously impractical hardware implementation that is useful for a conceptual understanding of sample-rate conversion. The input signal is sampled, stuffed with $2^{16} - 1$ zeroes between each sample, and applied to an FIR interpolation filter which is shown as a classic convolution machine employing a shift register in which each tap is multiplied by the corresponding coefficient value and summed to form the output. For now consider that the shift register is operated at the ($2^{16} \times 50$ kHz) 3.27 GHz rate and that a new interpolated output is produced on every cycle. The asynchronous output resampling switch then selects the "nearest" interpolated output when it closes. The interpolated output is held in a register for the duration of one cycle of the interpolation clock, resulting in a "free" zero-order-hold function.

To estimate the required filter length, a commercial filter design program was used to design a 20-kHz low-pass filter at sampling rate of 88.2 kHz, with a passband ripple of .01 dB and a stopband attenuation of 110 dB. The estimated length for such a filter is 128 taps. For a filter running at a sampling rate of $Fs = 2^{16} \times 44.1$ kHz, the estimated length is $128\times(2^{16}\times 44.1\text{ kHz})/88.2\text{ kHz} = 4\ 194\ 304$. This number also gives the shift register length and the number of stored coefficients. We have used a sample rate of $2^{16} \times 44.1$ kHz with a transition band from 20 to 24 kHz as the basis for this calculation, as this allows resampling at a 44.1 kHz rate with no signal image aliasing in the audio band.

B. Reduction to a Practical Hardware Implementation

Referring again to the conceptual hardware model of Fig. 4, the number of nonzero data values that exist at any one time in the shift register is the number of taps divided by 2^{16} which is 64. A further reduction in complexity can be made by realizing that there is no reason to compute every interpolated output at the 3.27 GHz rate when the vast majority of the computed output samples will be discarded by the resampling process. Together, these two observations lead to a solution that is quite realizable in a modern CMOS process. Since filter convolution need only be done upon the arrival of an output sample clock (Fs_{out}), the required computation rate is the product of Fs_{out} and the number of nonzero data values in the shift register at any one time (64). For Fs_{out} equal to 50 kHz, the time alloted for one multiply and accumulate (MAC) operation is about 180 ns which is quite feasible in a modern CMOS process.

The algorithm suggested above implies that the exact arrival time of an Fs_{out} clock must be measured, and this information is used to determine the location of the nonzero data values in the shift register. Once these locations are known, the correct subset of filter coefficients can be selected to multiply the data values presently in the shift register. In fact, the zero-valued data in the shift register would not have to be stored at all. Instead, a much shorter shift register could be used to store the input data (without zero-stuffing), and the correct subset of coefficients could be selected to convolve with the stored data. This process is seen to be a time-varying FIR filter of length 64. Depending on the relative phases of the input sample clock and output sample clock, a particular set of 64 coefficients out of a total coefficient space of 64×2^{16} can be chosen to compute the requested output. These subsets of filter coefficients are often referred to as "polyphase filters" in the

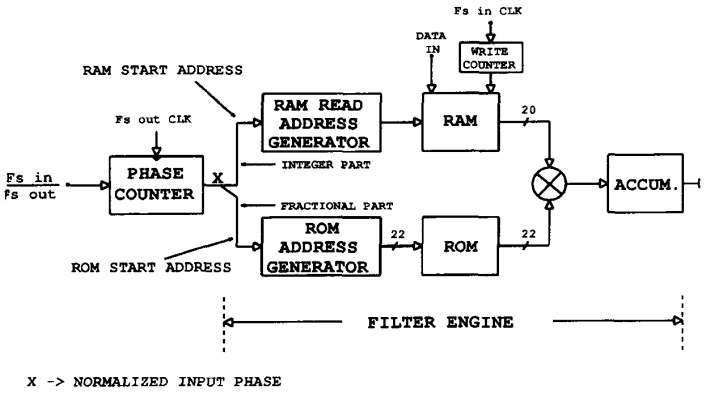

Fig. 5. Fixed sample-rate conversion given a known input over output frequency.

literature [5]. In summary, there are 2^{16} subsets of coefficients or polyphase filters in total.

The biggest challenge with this approach is measuring the arrival time of the output sample request with sufficient accuracy. The position of the nonzero data values in the shift register must be determined with no error when the output clock arrives, which implies that the arrival time of this clock must be measured with an accuracy of 1/3.27 GHz, or less than 300 ps. Such a precise measurement requires a master clock signal of 3.27 GHz, which is clearly undesirable. A solution to this problem is to average many coarse measurements in a way that the DC error is guaranteed to go to zero over the long term. A technique for performing this averaging will be discussed in the following sections.

C. Hardware for Synchronous Sample-Rate Conversion

Given a particular ratio of sample-rate conversion, a sequence of desired sampling points can be generated by putting the conversion ratio number $Fs_{\text{in}}/Fs_{\text{out}}$, assumed to be in binary format with a radix separator between the integer and the fractional part, into an integrator that periodically overflows, as shown by the "phase counter" block in Fig. 5. The fractional part of this integrator output points to the desired subset of filter coefficients (polyphase filter), and the integer part points to the data record (RAM). Convolution is then performed by stepping backwards through the RAM while multiplying these data values by the coefficients of the selected polyphase filter, stored in ROM. For the case of $Fs_{\text{in}} > Fs_{\text{out}}$, the pointer into the RAM will occasionally increment by 2, while for the case of $Fs_{\text{in}} < Fs_{\text{out}}$, the data record pointer will occasionally not increment at all. In either case, the total delay is constant because any required fractional delay is provided by the selection of an appropriate polyphase filter. Since a polyphase filter has a fractional delay that is proportional to their index [5], the fractional part of the phase counter can be used to directly address the appropriate polyphase filter. This technique can be used to perform sample-rate conversion on a set of stored data (for example, a sound file on a computer disc) by a fixed ratio. For asynchronous sample-rate conversion, the "ratio" number ($Fs_{\text{in}}/Fs_{\text{out}}$) that drives the modulo integrator must be measured from user-supplied clock signals.

D. Asynchronous Sample-Rate Conversion

In the last section we showed how to perform sample-rate conversion on stored data, given a desired ratio of input to output sample frequencies. This section shows how this ratio can be determined by using a servo loop which allows for the automatic tracking of asynchronous and dynamically varying sample rates. Fig. 6 shows a block diagram of a servo loop that can track the input-over-output frequency ratio. There are two input clocks: a "Fs_{in}" clock which is synchronized to the input word rate and an "Fs_{out}" clock which requests output data at the output sample rate. The operation of the ratio tracking loop can be explained as follows. When the estimate of the ratio of the input and output frequencies $\left(\widehat{\frac{Fs_{\text{in}}}{Fs_{\text{out}}}}\right)$ at the input to the phase counter is accurate, the phase counter will advance by $Fs_{\text{in}}/Fs_{\text{out}}$ on every output clock cycle. Over a fixed time interval T, the change in the phase-counter output is $\widehat{\frac{Fs_{\text{in}}}{Fs_{\text{out}}}} \times T \times Fs_{\text{out}}$ which is equal to $Fs_{\text{in}} \times T$. Over the same time interval, a counter connected to the input clock also advances by $Fs_{\text{in}} \times T$. Thus, the average slope of the phase counter output (X) as a function of time is always the same as that of the input clock counter. The phase counter output (X) can therefore be subtracted from the output of a counter connected to the input clock (write counter), and the resulting number is used to update $\widehat{\frac{Fs_{\text{in}}}{Fs_{\text{out}}}}$. In Fig. 6, a simple gain K is used to connect the error signal to the phase counter input, but in general it is desirable to add additional filtering in the loop to better track the input rates with the smallest amount of jitter possible. Traditional feedback-system analysis may be used to determine the dynamic behavior of the loop as it settles to its steady-state value.

E. The Undersampled Case

One subtle aspect of sample-rate conversion is that when $Fs_{\text{out}} < Fs_{\text{in}}$, signals can appear at the input to the converter that are less than the Nyquist frequency of the input sample rate but greater than the Nyquist frequency of the output

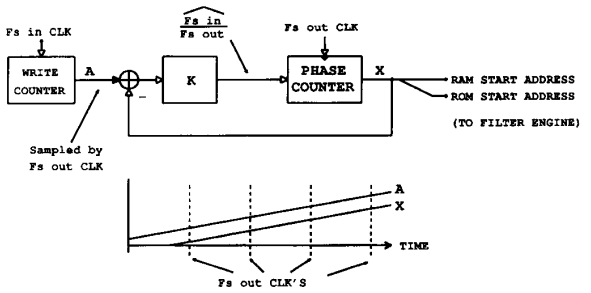

Fig. 6. A closed-loop method of estimating the ratio of input over output frequency.

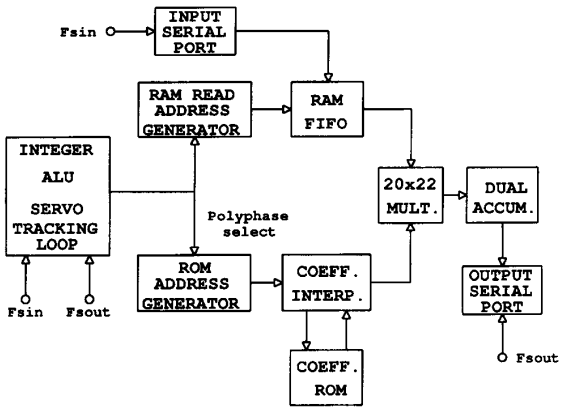

Fig. 7. A simplified block diagram of the internal architecture.

sample rate. Frequencies in this range will cause aliasing distortion to appear at the output unless precautions are taken. The preferred solution to this problem is to dynamically alter the stored interpolation coefficients such that the filter cutoff frequency is lowered as the output sample rate falls below the input sample rate. This can be accomplished by applying the scaling property of the Fourier transform to linearly expand or compress the frequency response of any filter; i.e., if the Fourier transform of $f(t)$ is $F(\omega)$, then the transform of $f(kt)$ is $F(\omega/k)$. In other words, if a signal is linearly compressed along the time axis in the time domain, then in the frequency domain the spectrum will be linearly expanded, and vice versa. Normally this theorem would not apply to discrete-time systems, because the new time steps that result after stretching or contraction are no longer aligned with the time grid corresponding to the original sampling frequency. For example, if we had a signal sampled at 1 MHz and we tried to multiply the time variable by 10%, the signal at time t = 8 μs should ideally come out at 8.8 μs, but since we are constrained to a fixed 1 μs time grid, this sample value would have to come out at 9 μs instead. This would ordinarily cause severe distortion, but in our case the time grid of the low-pass filter coefficients is so dense compared with the cutoff frequency (300 ps for a 20 kHz cutoff frequency) that the error introduced by rounding a contracted or expanded coefficient to the nearest 300 ps is negligible. The following steps are carried out to properly adjust the cutoff frequency: 1) The ROM addresses must be multiplied by a fractional number proportional to $Fs_{\text{out}}/Fs_{\text{in}}$. We will refer to this number as ROM-ADR-SCALE. 2) If $Fs_{\text{out}} > Fs_{\text{in}}$, ROM-ADR-SCALE should saturate at 1. That is, we do not want to alter the filter when the output sample rate is greater than the input sample rate. 3) If $Fs_{\text{out}} < Fs_{\text{in}}$, set ROM-ADR-SCALE to $Fs_{\text{out}}/Fs_{\text{in}}$.

While we already have the ratio of frequencies computed in the auto-ratio loop of Fig. 6, this is unfortunately the inverse of the ratio we need. A simple counter commonly employed in frequency meters to find the ratio between an unknown external clock and an internal high-precision reference frequency may be used to find the ratio $Fs_{\text{out}}/Fs_{\text{in}}$. Its measurement precision need not be very high, as a small error in the filter cutoff frequency is of little consequence in the final design. It is interesting to note that as the output sample rate is lowered, the filter length increases due to the smaller transition band of the interpolation filter characteristic relative to the fixed input clock frequency. Fortunately, this increase in the number of multiply/accumulates per output sample is exactly offset by the decreased frequency of the output request clock. This relationship allows the multiply/accumulate process to proceed at a constant rate that is independent of the output request rate. This argument does not apply to the case where the output rate is fixed while the input rate is increased. In this case, a faster multiply/accumulate cycle may be required, as the filter length is increasing while the available time in which to complete a filter calculation is fixed. Another consequence of the increasing filter length is that the RAM must be correspondingly larger to hold a larger number of past data values. This method of dynamically altering the filter cutoff frequency will only work for the "direct" sample-rate conversion method presented in this paper. Multi-rate techniques have been presented [1–4] where a fixed interpolation filter is followed by a short time-varying filter that computes the requested values between the time points outputed by the interpolator. A drawback of this technique is that the time-varying filter is too short to affect the 20 kHz frequency response of the combined filter. The only solution is to store multiple sets of interpolation filter coefficients for different ranges of sample-rate ratios, which lacks the simplicity and continuous fine adjustment capability of the algorithm proposed here.

F. ROM Reduction

To this point, we have assumed that the complete set of filter coefficients are stored in ROM. Unfortunately, due to the very high oversampling factor of this filter (at least 2^{16}), a very large number of coefficients must be stored (10 Mega-words of ROM for a typical filter). Reduction of the ROM size is important if a cost-effective VLSI chip is to be designed. Several methods may be used to reduce this to a more reasonable number. The simplest method is to use linear interpolation. Differences in adjacent ROM coefficient values are very small, due to the high ratio of the "sample rate" for which the filter was designed (3.27 GHz) and the cutoff frequency (20 kHz). Therefore, the ROM storage

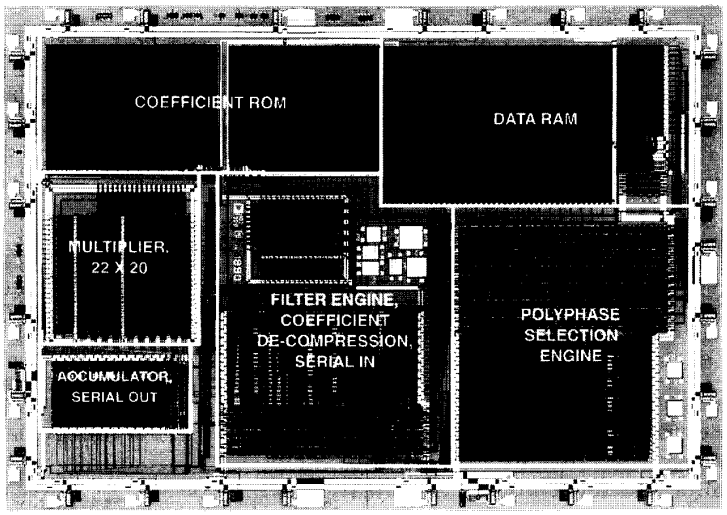

Fig. 8. Chip photomicrograph.

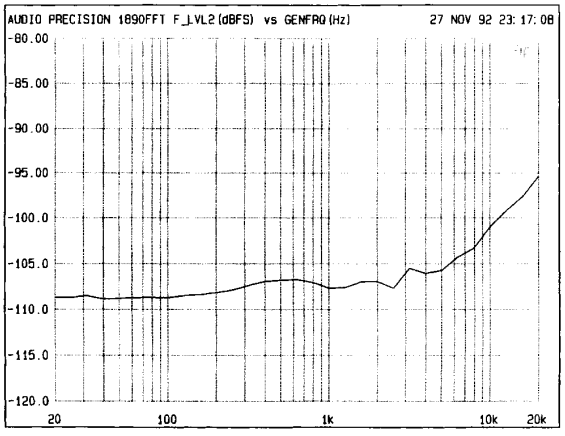

Fig. 9. THD + N versus frequency: $Fs_{\text{in}} = 48$ kHz, $Fs_{\text{out}} = 44.1$ kHz.

requirement can be reduced by storing only a sparse subset of the ROM coefficients and linearly interpolating between stored coefficients.

G. Chip Organization

Internally, the chip is divided into three major sections: a multiply/accumulate engine, an integer ALU for implementing the closed-loop ratio tracking circuit, and a computation block which implements the ROM coefficient interpolation. Fig. 7 shows a simplified block diagram of the internal architecture. The multiply and accumulate (MAC) engine uses 22-b coefficients operating on 20-b data. MAC's are performed at an 8 MHz rate, with left and right channel computations interleaved using a dual accumulator. The accumulator output is truncated to 24 b and passed through a saturation circuit to the serial output circuitry. The number of MAC's required to compute one output sample depends on the sample-rate ratio. In the case where the output sample rate is equal to or higher than the input sample rate, the filter length is 64 taps. For the undersampled case, the filter becomes correspondingly longer, as previously discussed. The ratio-tracking block is implemented in a general-purpose ALU with microcode stored in a program ROM. A data-path structure is used and includes such elements as a register file, adder/subtractor, shifters, etc. The serial input clocks supplied by the user are used to drive the loop integrator in the ALU as well as the RAM write address generator in the FIFO, as shown in Fig. 7. These clocks are synchronized to the 16-MHz master clock before they are used. Since these inputs are asynchronous to the master clock, special precautions are taken to prevent metastability problems from occurring. The output of the ratio-tracking block provides a starting address to both the ROM and RAM address generators, as well as an estimate of the out/in sample-rate ratio used for the automatic adjustment of the interpolation filter cutoff frequency. The filter coefficient-interpolation block is used to generate coefficients for the multiply/accumulate engine. There are 2 inputs to this block; the starting address (polyphase filter select) and the out/in ratio, which is used to scale all memory references so as to dynamically alter the filter response in the undersampled case. To simplify the system-level architecture, the coefficient interpolation block was designed to have a full 22-b input address bus. The details of how the coefficients are generated from a sparse set of stored coefficients is hidden from the rest of the system. The ROM address generator receives the starting address and out/in ratio from the integer ALU block and issues 22-b addresses accordingly. The top bits of this address field are used to access the physical ROM, while the bottom bits are used for linear interpolation. A small hardware multiplier is used for the linear interpolation function.

The control block consists of a sequencer which is idle until an output sample clock arrives. When the sequencer has completed the computation of an output sample, it is idle until the next output clock. Meanwhile, whenever an input sample

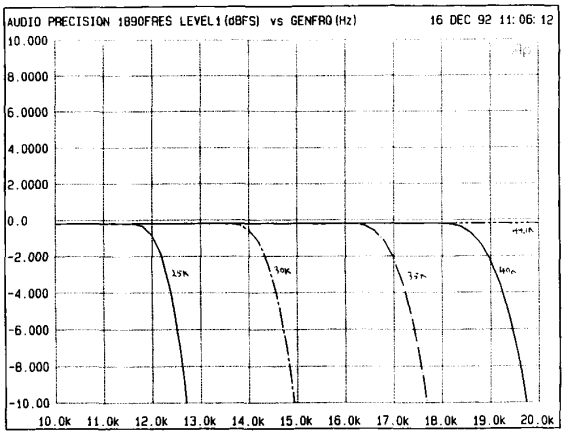

Fig. 10. Filter transfer function for $Fs_{\text{in}} = 44.1$ kHz, $Fs_{\text{out}} = 44.1$ kHz, 40 kHz, 35 kHz, 30 kHz, 25 kHz.

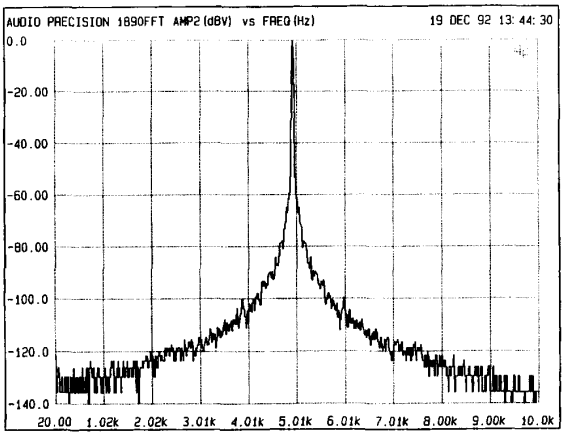

Fig. 11. FFT of output with 100-ns (rms) jitter on the input clock.

arrives, the MAC operations are interrupted so that the input data can be written to RAM.

IV. Measured Performance

A photomicrograph of the sample-rate converter is shown in Fig. 8. A chip summary is given in Table I. Initial silicon was measured with an Audio Precision System One. Fig. 9 shows a plot of THD + N versus frequency for a full-level signal. Fig. 10 shows the automatic adjustment of the filter cutoff frequency for the undersampled case. This was done by reducing the output sample rate in steps of 5 kHz from a 44.1 kHz rate, with a fixed input rate of 44.1 kHz. In general, the measured results showed excellent agreement with simulations. For frequencies below 5 kHz, THD + N figures are about 107 dB, degrading to about 102 dB with a full-level 10 kHz input and 96 dB with a full-level 20 kHz input. This is in good agreement with analysis presented in the previous section that suggested that 16-b worst case performance could be achieved with a time resolution of 1 part in 2^{16} between input samples. Since the amplitude errors incurred by the resampling process are directly proportional to the input slew-rate of the input signal which is in turn proportional to the input amplitude, the ratio of signal strength to distortion is independent of the level of the input signal. This implies that as the signal level is reduced, the distortion and noise components are also reduced proportionally. This scaling of distortion with input level gives the device a true 20-b dynamic range.

One benefit of using this type of sample-rate conversion is that jitter on the input or output clocks is heavily filtered. Fig. 11 shows the spectrum of a sine wave in which the input clock has been jittered by a binomial noise source with a RMS jitter value of 100 ns[1]. This is obviously an extreme case, chosen to make the artifacts more noticeable. Note that the jitter produces narrow-band noise skirts around the fundamental frequency, which would likely be inaudible due to psychoacoustic masking effects.

[1] A full level 20-kHz sine wave with 100-ns jitter has a signal-to-noise ratio of approximately 41 dB.

TABLE I
Chip Summary

Technology	0.8 μm CMOS
Chip size	5.2 mm × 7.2 mm
Transistor count	210 k
Power dissipation	51 mW @ 3 V
	142 mW @ 5 V
Operating frequency @ 5 V	0-20 MHz
Output frequency	5–56 kHz @ MCLK=16 MHz

Listening tests have been performed with special attention given to dynamic effects while tracking real-time sample-rate variations. No artifacts have been noted during these tests, except for the unavoidable brief doppler-shift effect for sample rates that change much faster than the time constant of the ratio-tracking loop.

V. Conclusion

A topology for asynchronous sample-rate conversion has been presented which is well suited to VLSI implementation. It achieves 16-b performance (worst case full-level 20 kHz input) with modest amounts of RAM, ROM, and digital filter hardware. The proposed algorithm uses an unique auto-ratio servo loop that can track real-time sample-rate changes with no discontinuities and reject high-frequency jitter greater than 3 Hz. It can also dynamically alter the filter cutoff frequency when the output sample rate falls below the input rate to prevent aliasing. A VLSI implementation of this algorithm has been disclosed along with measured results that closely match the theory. This chip should largely solve the problem of digital-audio connectivity, making it possible to connect different pieces of equipment together in the digital domain without having to resolve difficult synchronization issues.

Acknowledgment

The authors would like to thank M. Coln for contributing to the architecture of this chip, T. Tsang who expertly designed some of the datapath cells, and the reviewers, whose comments improved the presentation of this paper.

REFERENCES

[1] G. W. McNally, R. Lagadec, D. P. Pelloni, "Method and apparatus for measuring the time difference between two sampling times," U.S. Patent 4 564 918.
[2] A. Koch, R. Lagadec, and D. Pelloni, "Method and Apparatus for Converting an Input Scanning Sequence into an Output Scanning Sequence," U.S. Patent 4 825 398.
[3] R. Lagadec,"Scanning frequency synchronization method and apparatus," U.S. Patent 4 780 892.
[4] R. Lagadec and H. Kunz,"Process and apparatus for translating the sampling rate of a sampling sequence," U.S. Patent 4 748 578.
[5] R. Crochiere and L. Rabiner, *Multirate Digital Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1983.

Robert Adams received the B.S.E.E. degree from Tufts University in 1976.

In 1978 he joined the staff of dbx Inc., and in 1982 became Director of Audio Research. While at dbx, he designed a variety of products including an early audio digital recorder based on companded predictive delta modulation, and later a standalone 18–b sigma–delta A/D/ converter. In 1989 he joined the staff of Analog Devices, Inc., as Manager of Audio Technology. Since then he has been engaged in the design of sigma–delta A/D and D/A converters, and most recently led the design of an all-digital asynchronous sample-rate converter chip.

Mr. Adams is a fellow of the AES and holds many patents in the area of audio signal processing.

Tom Kwan received the B.A.Sc. degree in engineering science from the University of Toronto, Canada, in 1984, and the M.S. and Ph.D. degrees in electrical engineering from the University of California, Los Angeles, in 1986 and 1990, respectively.

Since 1990, he has been with Analog Devices, Norwood, MA, working in the area of mixed-signal audio products. His technical interests include analog and digital filter design, adaptive signal processing, and oversampled data converters.