THE EMU10K1 DIGITAL AUDIO PROCESSOR

THIS PC AUDIO SOLUTION FULFILLS ITS ORIGINAL DESIGN GOAL AS A MICROSOFT DIRECTSOUND ACCELERATOR AND, WITH ITS ENVIRONMENTAL SIMULATION CAPABILITIES, PROMPTED THE DEVELOPMENT OF THE ENVIRONMENTAL AUDIO EXTENSIONS (EAX) TO MICROSOFT DIRECTSOUND3D. ORIGINALLY DESIGNED FOR THE CONSUMER COMPUTER MARKET, THE EMU10K1 ALSO SUPPORTS PROFESSIONAL AUDIO CONTENT DEVELOPMENT.

Thomas C. Savell
Joint Emu/Creative Technology Center

Consumers and professional audio content developers continue to demand more powerful audio systems in the virtual worlds of games. These needs led our company to produce the EMU10K1 digital audio processor, which places both a high-quality music synthesizer and a powerful audio effects processor on the same die.

Implemented in a 0.35-micron, three-metal-layer CMOS process, the processor resides on a 6.7-mm × 6.5-mm die containing 2,439,711 transistors, a 33-MHz PCI clock, and 50-MHz and 100-MHz internal audio clocks. It has a 64-channel wavetable synthesizer with per-channel sample rate converter, digital filter, envelope generator, low-frequency oscillator, and routing/mixing logic. The EMU10K1 accesses digital audio data stored in system memory via a 64-channel PCI bus master system with virtual memory mapping identical to that in Intel CPUs. A powerful audio effects processor adds environmental simulation, 3D positioning, and special effects to audio from the wavetable synthesizer and numerous other digital audio sources such as S/PDIF (Sony/Philips Digital Interface), I²S (Philips Inter-IC Sound bus), and AC97 (Audio Codec 97). Since the processor operates at a fixed 48-kHz audio sampling rate, it contains dedicated, high-quality sample rate converters.

PC audio subsystem

This subsystem has evolved to be quite complex with numerous methods for software applications to interact with various audio sources and destinations. Applications can either call the operating system to play back audio stored in disk files or directly place the digital audio waveforms in system memory and make device driver calls for playback. Applications can produce music using an abstract language known as MIDI[1] that supports commands such as "play middle-C" and "use grand piano sound." MIDI commands can even originate from an external synthesizer or keyboard controller connected via a cable to the computer. In addition to these sources, other devices such as CD-ROM, DVD, and microphones also produce audio that the user can hear and record to a disk file. Recent 3D computer games tend to include 3D audio as well as graphics, requiring additional processing to create a virtual 3D audio environment.

Architecture

The major architectural elements of the EMU10K1 are the PCI bus master and slave

0272-1732/99/$10.00 © 1999 IEEE

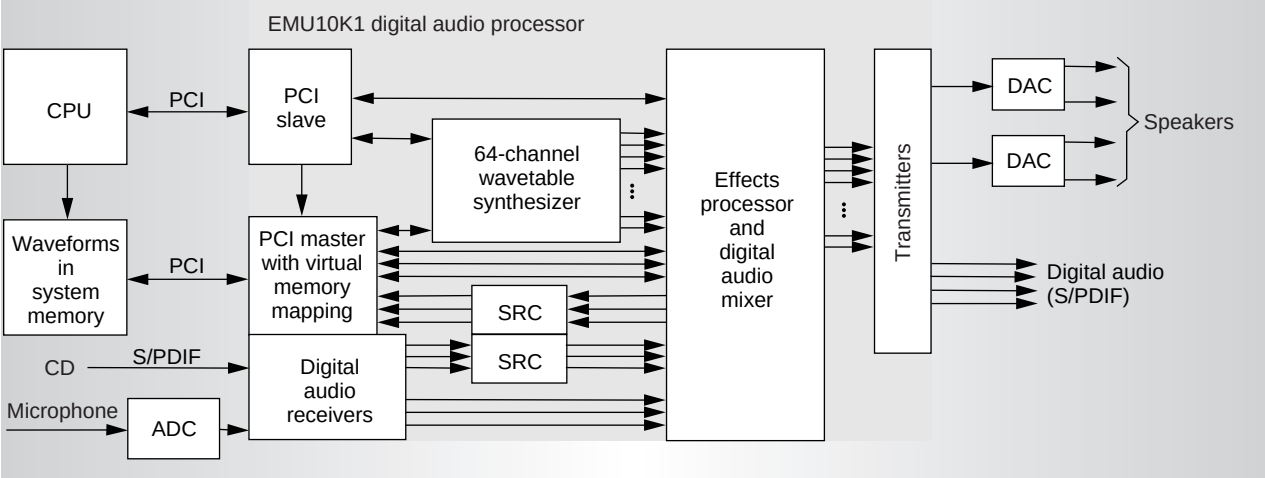

Figure 1. PC audio subsystem. All audio sources pass through the effects processor where they are mixed together to create a virtual 3D environment for rendering on a multiple-speaker system.

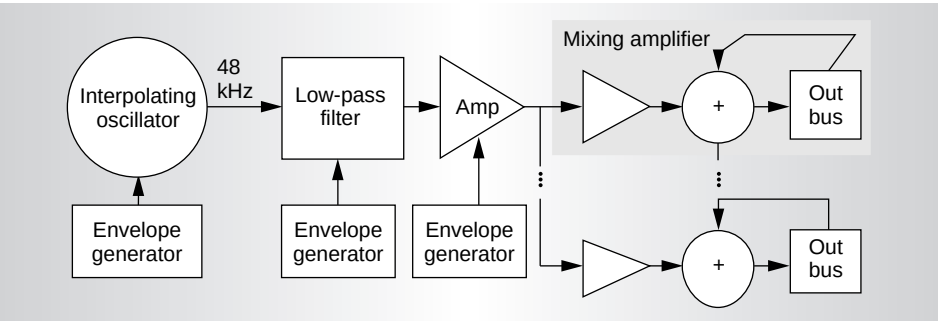

Figure 2. EMU10K1 wavetable synthesizer.

interfaces, 64-channel wavetable synthesizer, effects processor, high-quality sample rate converters, and digital audio receivers and transmitters. Figure 1 illustrates the connection between blocks and their interaction with the application environment.

Wavetable synthesizer

Each of the 64 channels in the wavetable synthesizer consists of an interpolating oscillator; a resonant two-pole, digital low-pass filter; three envelope generators; and a mixing amplifier with multiple selectable output buses. See Figure 2.

Background. Wavetable synthesis uses a digital recording of a natural sound such as a note played on a piano. The wavetable synthesizer replays this recording in response to a player striking keys on a keyboard. When synthesizing an instrument such as a piano, it is not necessary to record every note the instrument can play. To conserve memory, the sound designer samples a small number of notes from across the entire range of the instrument, and the synthesizer creates intermediate notes on the fly with pitch-shifting hardware. The equivalence of pitch shifting and sample rate conversion allows the user freedom in choosing the source sample rate. The wavetable synthesizer includes looping hardware to create notes of arbitrary duration and further conserve sample memory. It responds to the player's touch with greater expressiveness, using the envelope generator to control the amplitude and filter cutoff.

Interpolating oscillator. The EMU10K1 wavetable synthesis oscillator contains an interpolating sample rate converter and an addressing unit capable of looping and proceeding at an arbitrary, noninteger rate. The addressing unit maintains a phase accumulator that contains the current memory address, stored in an integer.fraction format. The integer portion determines the memory read locations, and the fractional portion determines the interpolation phase shift. The addressing unit adds a phase increment, stored in the same integer.fraction format, to the current address every sample period.

The looping hardware uses two programmable addresses, the loop start address and the loop end address. When the waveform address passes the loop end address, the addressing unit loads the current address with a value equal to the loop start address plus the amount by which the address would have passed the loop end address. This causes the fetching of data at the loop start address again and repeating the loop until stopping the oscillator. Note that the current address register can accept any value to start with, and it proceeds without interruption until it reaches or passes the loop end address. This accommodates a different, nonrepeating waveform during the initial attack of a note, while providing an effective form of data compression and placing no predetermined limit on the length of time a sound can play.

The 8-point interpolating sample rate converter uses the fractional address to determine the position between input samples at which to output a new sample. This is performed using the well known Smith-Gossett[2] algorithm, which requires 16 multiplies and 24 adds. An on-chip ROM stores the filter coefficients used in the interpolation algorithm. The EMU10K1 uses extended-precision arithmetic to tolerate intermediate overflow operations and to detect output overflow. In the case of output overflow, the sample rate converter produces a saturated output, avoiding severe two's-complement wraparound distortion.

Resonant digital filter. The resonant digital filter implementation is a two-pole low-pass design. Figure 3 shows the filter topology, which requires two storage elements, three multiplies, and four adds. The EMU10K1 uses techniques presented by Rossum[3] in 1991 to reduce quantization noise, increase the pole angle resolution at low frequencies, and increase the pole radius resolution near the unit circle. The implementation of these techniques requires special encoding of the coefficients and an additional two's-complement operation.

The filter is capable of more than 20 dB of resonance. The envelope generator can sweep the filter cutoff frequency in real time.

Envelope generator. Traditional envelope generators for music synthesizers contain four phases: attack, decay, sustain, and release. The envelope generator of the EMU10K1 actually has six phases: delay, attack, hold, decay, sustain, and release, as illustrated in Figure 4. The delay and hold phases provide additional control to the sound designer.

Simple playback of recorded samples does not have the expressiveness of a musician playing a real instrument. This is because there are numerous ways the instrumental sound changes in response to a musician's playing style. When the musician strikes an instrument harder, the sound is louder, has a sharper attack, and is brighter. A softer stroke results in a quieter sound, a more gradual attack, and is duller. The envelope generator of the EMU10K1 produces these effects by separately controlling the amplifier gain and filter cutoff frequency.

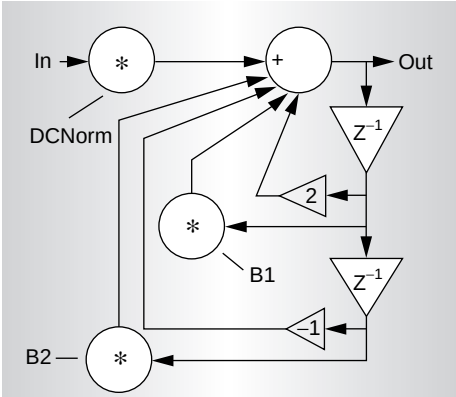

Figure 3. Digital filter topology.

Virtual memory mapping

Within the PC environment, application memory is virtualized into 4-Kbyte pages that are mapped into a larger contiguous address space. An application program must request an allocation of memory, a buffer, from the operating system. Throughout the course of normal computer usage, programs allocate and deallocate memory buffers millions of times. As the process continues, it may become impossible to find a single block of physical memory to satisfy an allocation request. The operating system uses virtual memory mapping to solve this problem. Each buffer of memory, while addressed as a contiguous whole, may in fact be fragmented into many 4-Kbyte pieces randomly scattered throughout the physical memory. Since a DMA bus master must present physical addresses to the bus, there must be a way to resolve the virtual addresses used by the application program.

Double buffering. One solution is to allocate a single, physically contiguous buffer into which audio is copied from fragmented virtual memory prior to DMA transport. Due to operating system limitations, physically contiguous buffer allocation is not guaranteed, and it can only be requested during system start-up. Even if it is successful, the

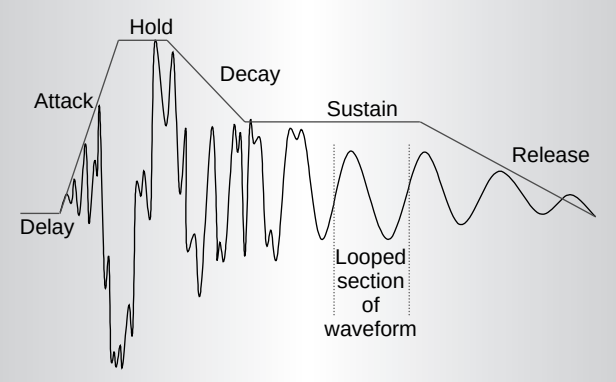

Figure 4. The various phases of the EMU10K1 envelope generator overlaid on the digital audio waveform. The attack phase typically contains more high-frequency information as well as wideband noise. The high frequencies and noise content become attenuated during the decay phase, and a fundamental oscillation, or pitch, becomes prevalent. Once this occurs, a few points that are looped during the sustain and release phases can represent the remainder of the sound.

memory cannot be dynamically allocated and deallocated at runtime. This is a decided disadvantage to the user, as a large segment of main memory must be permanently assigned as an audio buffer. It also incurs the CPU overhead of copying buffers from virtualized memory to the physically contiguous buffer.

Scatter-gather. Another way is to pass a scatter-gather list to the stream transport engine. A scatter-gather list is an ordered set of physical addresses that must be parsed as the stream is being transported. This allows dynamic allocation and deallocation, and does not require the copying of buffers. However, the same sound is often triggered multiple times simultaneously, and for music and game applications each instance may require different sample rate conversion ratios. This requires redundant copies of the same scatter-gather list. Even more significantly, sounds that loop to allow them to play for long periods require excessively long lists.

Page table. The EMU10K1 uses a better method, translating the addresses in a similar fashion as the CPU by using a page table and a translation look-aside buffer. The page table is located in system memory and contains the physical addresses of each 4-Kbyte page within the logical sound memory, as shown in Figure 5 (next page).

The on-chip translation look-aside buffer contains the physical mapping of the current logical address. As the stream transport engine moves through logical memory, it detects when the logical to physical mapping is no longer valid, and issues a read from the page table to reload the translation look-aside buffer. This method is very efficient for both the CPU and audio stream transport engine. It permits dynamic allocation of audio buffers and has a minimum of redundancy.

As wavetable oscillators proceed, they submit PCI bus master requests for data. A two-level priority scheme based on the degree of data starvation arbitrates bus master access. This reduces the probability that an oscillator will run out of input samples. A single bad sample can be audible, especially when inputting the audio to a recursive delay line.

Effects processor

The EMU10K1 effects processor creates the final output that the listener will hear on a multiple-speaker system. It satisfies the mixing requirements of the PC environment with a total of 31 inputs and 32 outputs. The 24-bit audio I/O capability provides an ample 144-dB dynamic range, enough to span the entire human hearing range from perceived silence to past the threshold of eardrum damage. However, all instructions use 32-bit integer or fixed-point operands to support the additional precision required for complex operations such as recursive filtering.

ALU. At the core of most signal-processing algorithms is the multiply-accumulate operation. Naturally, the center of the EMU10K1 effects processor is a high-speed arithmetic logic unit that implements variants of this powerful operation. All instructions use four register addresses: result, accumulator, multiplier, and multiplicand. For maximum flexibility, the operand address space is symmetrical in that any address is valid for use in any operand. The effects processor forwards the result register to conceal pipeline delays and permits the result to be used as an input operand on the next instruction period. It processes 32-bit fixed-point and integer operands and has a 67-bit accumulator to allow for intermediate overflows during multistage operations such as

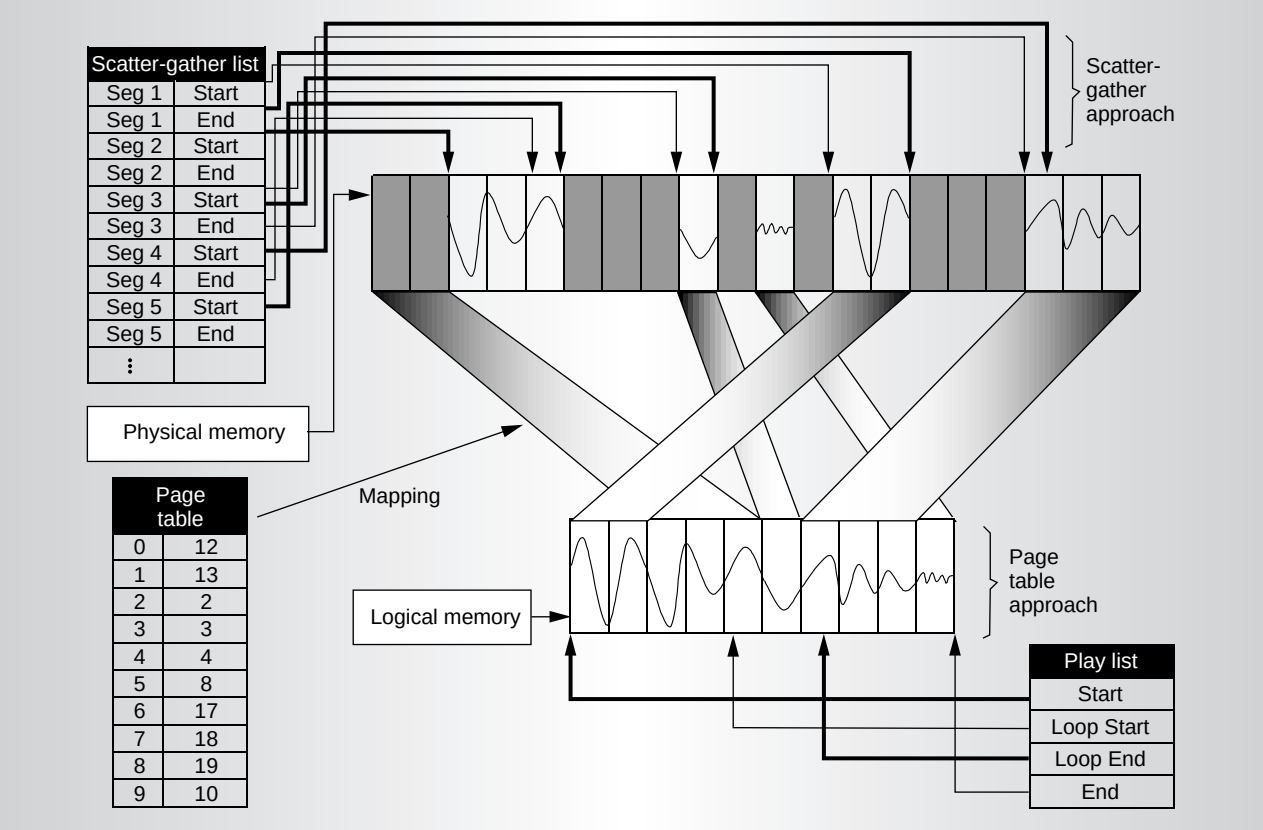

Figure 5. Scatter-gather (upper) versus page table (lower) virtual memory approaches. Both approaches render the same waveform. However, scatter-gather can require a very long list to support looping, while the page table requires no extra data.

finite-length impulse response filtering.

Delay lines. Delay line buffer registers provide zero-latency access to a small on-chip delay memory and a large delay memory located in the PC host memory. A dedicated delay line processor performs the tedious job of maintaining all the circular addresses. The microcode works with logical offset addresses that do not need updates on every sample period, except for special effects that require modulation of the delay length. Still, only the modulation requires microcode for calculation; the delay line processor automatically performs the circular address increment operation.

A dedicated data movement engine absorbs the long latency of a large external memory. The effects processor requires instant access to the delay line. That is not possible when the delay memory is physically located across an arbitrated bus such as the PCI. Access time can even present a challenge to a local memory, since delay memories must be many thousands of bytes deep to be useful. By dedicating a small buffer memory to store the "current" data for each delay pointer, access time can be instantaneous. The data movement engine then can spread out the accesses into the large delay memory over the course of a sample period.

Instruction sequencing. The effects processor has a novel architecture: it does not contain a program counter or support branch/loop constructs. Instead, a powerful conditional execution mechanism provisionally stores the results of operations. A special SKIP instruction that implements the following logic accomplishes conditional execution:

```
IF (cond) THEN SKIP instruction_count
```

The conditional expression, cond, is a Boolean AND-OR-NOT combination of a mask operand with a hardware register that stores condition codes from previous instructions. This supports both conventional expres-

sions such as less-than and more complex conditions such as negative-and-overflow-or-zero. The execution of a SKIP instruction does not alter the order in which instructions are fetched. Rather, the processor still fetches, decodes, and executes skipped instructions, but does not write back the results.

The processor determines the number of consecutive instructions to be skipped from the instruction_count operand. This permits skipping entire blocks of code and facilitates real-time loading and unloading of signal-processing programs without affecting audio output. The SKIP instruction even serves as the NOP instruction, using the construct, ALWAYS SKIP ZERO, which does not store results and continues executing the program on the next instruction cycle.

For every output sample period, the effects processor reads all microcode memory in sequential order and skips the writing of result registers based on the conditions specified in SKIP instructions. This provides for if-then-else execution, but not looping. A very important side effect of this architecture is that the total execution time of all concurrent programs is always exactly one sample period.

In a general-purpose DSP, one must take care to ensure that the entire program executes within a sample period. Otherwise, audible defects can result. In the case of infinite-length impulse response filters and recursive delay lines, a single sample defect can remain audible for a very long time. Strange bugs in audio-processing algorithms have been traced to an occasional interrupt service routine that caused program execution time to exceed the length of a sample period. These types of bugs are not possible in the EMU10K1 effects processor.

Asynchronous digital audio receivers

We designed the EMU10K1 to receive digital audio directly from devices such as CD-ROM and DVD drives. However, the sample rate of compact disc audio is 44.1 kHz, and the EMU10K1 output sample rate is 48 kHz. Due to manufacturing tolerances and drift, the clock frequency of each compact disc player differs slightly. Even if the output sample rate were also 44.1 kHz, the slight differences in clock frequency would cause the relative phases of the input and output sample rates to drift over time, eventually resulting in repeated or dropped samples.

It is possible to force the clock frequencies to be exactly synchronous by using a tracking phase-locked loop, or PLL, rather than a fixed-frequency oscillator. Professional recording studios distribute a master clock to all interconnected digital audio devices, which derive local clocks from the master. This guarantees synchronicity of all digital audio streams. This approach is expensive and difficult, requiring PLL-based synchronization capabilities in all digital audio devices. Devices that cannot synchronize to an external clock source must become the master clock source. This is a distinct disadvantage as there can only be one master clock at a time.

A better solution is to use sample rate conversion to resolve the incoming sample rate to the output rate. This requires a sample rate detector that continuously updates an estimate of the asynchronous digital input rate. The sample rate estimate maintains a phase accumulator that controls a 16-point Smith-Gossett[2] sample rate converter. Such an asynchronous sample rate converter avoids the cost of a tracking PLL and provides support for multiple, simultaneous asynchronous audio streams. The EMU10K1 can support three simultaneous asynchronous stereo streams using high-quality asynchronous sample rate conversion.

Digital audio recording

Speech recognition, Internet telephony, music recording, and content authoring applications need digital audio recording capabilities. Noncritical recordings can use reduced sample rates to decrease data storage requirements. The standard kHz rates needed within the PC are 48, 44.1, 32, 24, 22.05, 16, 11.025, and 8. The conventional method is to reduce the clock rate of an analog-to-digital converter, or ADC. However, this requires the use of separate ADC and DAC chips rather than an inexpensive monolithic codec.

To ensure adequate alias rejection at all sample rates, more expensive tracking analog filters must be used as well. The EMU10K1 supports the various recording sample rates with very high-quality, 64-point sample rate converters, thus permitting the use of low-cost, monolithic codec chips and simple analog antialiasing filters. Recording incurs very low CPU overhead by using bus master DMA directly to sys-

Digital audio mixing

Audio mixing is a weighted summation of the inputs with saturation to avoid the severe distortion caused by two's-complement overflow. Mixing multiple digital audio streams requires that the streams have exactly the same sample rate. However, CD audio sampled at 44.1 kHz will inevitably need to be mixed with voice or sound effects sampled at some other rate. Lower sampling rates provide an effective data compression method for sounds that do not have substantial high-frequency content. In addition, the native sample rate of the digital audio output may be different, typically 48 kHz. The solution is to convert all sound sources to the output sample rate before mixing, as shown in Figure A.

Digital audio is simply a numeric representation of an analog waveform. There are infinitely many digital representations of the same analog waveform. Sample rate conversion is a way to transform one representation into another, which can be done with various degrees of quality and corresponding complexity. To create a new sample stream at a different rate, one must interpolate between the original samples. Figure B illustrates the three commonly used forms of interpolation: nearest-neighbor, linear, and multipoint.

The simplest form, nearest-neighbor interpolation, requires no arithmetic and has the worst quality. A better method is linear interpolation, which requires a small amount of arithmetic and has reasonable quality. Multipoint interpolation requires significant arithmetic, a coefficients table, and multiple input samples to create a single output sample,[1] but produces the best quality.

We can view sample rate conversion as a three-stage process. First, the conversion algorithm inserts a number of intermediate zero-valued samples in between the original samples, thus creating a new sample stream at a higher sample rate. The new rate is an integer multiple of the original rate and is typically thousands of times higher when converting arbitrary sample rates. Then, the algorithm filters the new high-rate stream to the lower of the input and output Nyquist frequencies. The stopband of the low-pass filter must sufficiently reject aliases to achieve the desired audio quality. Finally, the low-pass filtered stream is decimated to generate output samples at the desired rate.

High-order sample rate conversion is an extremely computationally intensive operation. To maintain equivalent quality from input to output, noise due to aliasing must be below the magnitude of the least significant bit on the input signal. For example, a 20-bit digital audio waveform has a dynamic range of about 120 decibels. To get equivalent 20-bit output, noise introduced in the sample rate conversion process must be less than –120 dB. For a 20-bit stereo signal, this requires more than 320 MIPS of processing power.

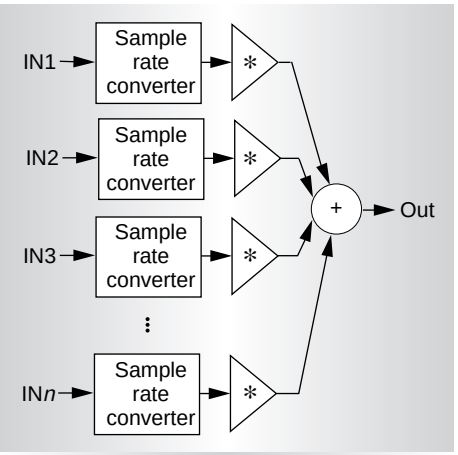

Figure A. Digital audio mixing.

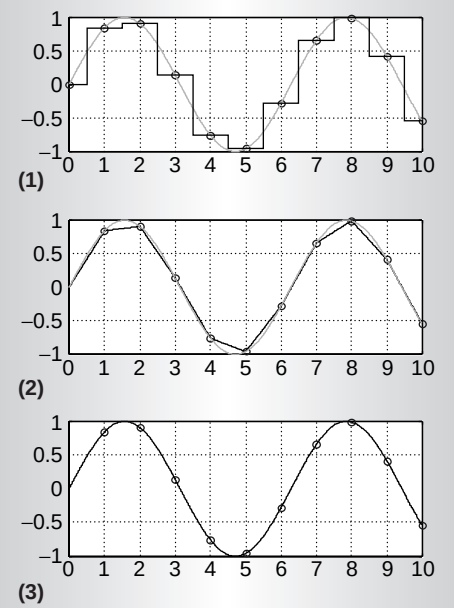

Figure B. Interpolation methods: nearest-neighbor (1), linear (2), and ideal multipoint (3).

References

1. R.E. Crochiere and L.R. Rabiner, *Multirate Digital Signal Processing*, Prentice-Hall, Upper Saddle River, N.J., 1983, pp. 127-190.

tem RAM. At the half-buffer and full-buffer points, an interrupt signals that software should flush the audio to disk. All 32 output channels of the effects processor may be selected for multichannel recording, enabling the use of the EMU10K1 in a home studio environment.

Sample rate conversion

The EMU10K1 achieves relatively high-order multipoint conversion using Smith and Gossett's particularly efficient algorithm of linearly interpolating the filter coefficients for convolution with the sample data stream. Rather than incurring the high cost of ideal conversion, the EMU10K1 uses a perceptual optimization technique so that most distortion components are inaudible. The key discovery was that most of the energy in real musical sounds is in the low-frequency band of human hearing. The images of these low-frequency components are located near multiples of the sample rate.

Designing the antialiasing filters to have deep

notches at multiples of the sample rate at the expense of some additional high-frequency aliasing significantly improves the perceived sound quality results. Rossum[4,5] received a patent in 1992 for interpolation filters designed in this manner. The high-frequency alias components do not have a great deal of energy because the source material is music, which has very little information in the upper band. In addition, Fletcher and Munson[6] discovered that human hearing is strongest in the middle frequency range and very weak in the extreme low- and high-frequency range. Thus, the human hearing system itself attenuates the small amount of high-frequency aliasing. These notches provide a dramatic improvement in the sound quality with no increase in computational complexity. Consequently, the EMU10K1 wavetable synthesizer has higher sound quality than would otherwise be expected when using only 8 points for interpolation. All EMU10K1 sample rate converters use the same technique.

Physical design characteristics

As stated earlier, the EMU10K1 is implemented in a 0.35-micron CMOS process with three metal layers. The 6.7 mm × 6.5 mm die with 2,439,711 transistors uses a 33-MHz PCI clock and internal audio clocks running at 50 and 100 MHz. The chip has 108 signal pins in a 144-pin PQFP. The device dissipates about one watt in normal operation, resulting in a worst-case junction temperature of 105°C, assuming ambient temperature of 70°C. The core operates at an internal voltage of 3.3 V, but the I/O is 5-V tolerant and supports both 3.3-V and 5-V PCI buses.

Design methodology

We used a VHDL-synthesis method to design the EMU10K1, although we first developed portions of the design using a C-language model and then translated them into VHDL.

Logic verification

Initially, we verified the individual blocks using RTL simulation. As chip-level integration progressed, we ran more simulations to verify the connections between blocks. As we got closer to tape-out, the importance of simulation faded and gave way to emulation. The design complexity coupled with the numerous asynchronous clock boundaries called for the use of emulation for final verification. Using the emulator, we could operate the design in an actual PC and run millions of times more cycles than would be possible using simulation alone. We could also sniff out a few nasty asynchronous boundary bugs that would have otherwise made it into silicon. Emulation does have its pitfalls, however; it is an immature technology that often left us feeling frustrated.

Because of the emulator's physical size and its FPGA use, the design cannot be run at full speed. To accomodate the low clock rate of the emulated design, we modified the target PC motherboard to reduce its PCI clock speed. We found that a factor of about 1:50 was necessary for reliable operation, so we operated the PCI bus at 0.6 MHz. We then ran the audio codec at the reduced rate, so we could observe analog audio output on an oscilloscope and even perform spectral analysis to detect distortion. These benefits far outweighed the inconveniences we suffered in getting the emulation environment up and running.

To make efficient use of the emulator, we required early software support to write a chip debugger that gave us direct access to the chip's features. The debugging software needed to be a DOS program, since Windows required over an hour to boot. The chip debugger supported macro scripts, so we could build higher level operations from sequences of commands. That made it easy to perform first-silicon evaluation by reusing the macro scripts written during development.

Timing analysis

We used static timing analysis for the majority of sign-offs. We could not properly analyze certain design sections, so we relied on a combination of SDF-annotated (Standard Delay Format) gate-level simulation and "correct-by-design" practices supported with billions of emulation cycles.

One of the more difficult challenges overcome during timing analysis was fixing setup and hold violations on the internal RAM inhibit pins. We chose to inhibit the RAMs on all unused clock cycles to conserve power, thereby reducing the worst-case junction temperature and increasing our chance of successful silicon. The vendor's RAMs inhibit pins worked by gating the clock. This required setup time before

the clock's falling edge and hold time after the clock's rising edge, reducing our timing window to much less than 5 ns in some cases. We solved the problem with careful placement and routing, and by tweaking the clock arrival times in layout. Also, satisfying critical PCI timing required careful placement of the PCI core and, in some cases, flip-flop duplication so the outputs could be placed close to the pad cell.

Testability

The chip has several test modes including scan, parametric NAND tree, and RAM test. To support extremely high production volumes, we used functional test vectors to supplement the scan test and achieve the highest possible fault coverage. We developed the test vectors by running an RTL simulation to record the pin states.

Interestingly, we ran a gate-level simulation to verify vectors. We used emulation as our primary logic verification tools. So the primary reason to run the gate-level simulation was to verify that the vectors would not fail because of RTL/gate differences in the modeling of unknowns (*X*'s). The vectors also provided handoff verification at the vendor. We encountered a number of vector mismatches during the handoff that were traced to inconsistencies in the handling of *X*'s by the internal SRAM models between the various simulators involved.

The EMU10K1 is an evolutionary step forward in the development of digital audio for the PC. The choice of placing both a high-quality music synthesizer and a powerful audio effects processor on the same die has helped to push the expectations for 3D audio in the PC.

The role of the 3D audio accelerator has become analogous to that of the 3D graphics accelerators. The demand for more power will continue as game developers create more elaborate virtual worlds that use up processing bandwidth faster than CPU manufacturers can create it.

The 33-MHz, 32-bit PCI bus appears to provide adequate audio bandwidth for now, but it could become a bottleneck in the near future. The EMU10K1 internal arbitration and priority scheme is very optimal and leaves very little room for increased memory access performance. Newer 0.25-micron and smaller processes present some challenges in terms of I/O voltage, especially considering that most PC motherboards have 5-V-only PCI buses. These system limitations will eventually need to be eliminated to allow further growth and innovation in the PC audio subsystem. MICRO

References

1. *The Complete MIDI 1.0 Detailed Specification,* MIDI Manufacturers Assoc., 1984-1996.
2. J.O. Smith and P. Gossett, "A Flexible Sampling-Rate Conversion Method*," Proc. IEEE Int'l Conf. Acoustics, Speech, and Signal Processing*, IEEE Press, Piscataway, N.J., Mar. 1984, pp. 19.4.1-19.4.4.
3. D. Rossum, "The Armadillo Coefficient Encoding Scheme for Digital Audio Filters," *Proc. IEEE ASSP Workshop on Applications of Signal Processing to Audio and Acoustics,* 1991.
4. D. Rossum, U.S. patent no. 5,111,727, May 12, 1992.
5. D. Rossum, "Constraint Based Audio Interpolators," *Proc. IEEE ASSP Workshop on Applications of Signal Processing to Audio and Acoustics,* 1993.
6. H. Fletcher and W.A. Munson, "Loudness, Definition, Measurement and Calculation," *J. Acoustic Soc. of America*, 1933, Vol. 6, p. 59.

Thomas C. Savell is a staff ASIC engineer at the Joint E-mu/Creative Technology Center. His responsibilities include digital and audio VLSI specification, architecture, design, implementation, and verification. He holds a BA in music technology from the University of California, San Diego, with a double minor in computer science and cognitive science. He is a member of the IEEE Computer Society, Signal Processing Society, and MIDI Manufacturers Association Technical Standards Board for 1997 and 1998.

Direct questions concerning this article to the author at the Joint Emu/Creative Technology Center, 1600 Green Hills Road, Suite 101, PO Box 660015, Scotts Valley, CA 95067-0015; tcs@emu.com.