An Audio Processor Card for Special Sound Effects

Fan Peck Ling, Fung Kah Khuen
School of Computer Engineering
Nanyang Technological University
Nanyang Ave., Singapore 639798

Damu Radhakrishnan
Department of Electrical and Computer Engineering
State University of New York, 75 South Manheim Blvd.
New Paltz, NY 12561

***Abstract*- The design of an audio processor card for generating special sound effects is presented in this paper. This is designed as an add-on card that plugs directly into the ISA bus of a PC. The card uses a Motorola DSP processor, DSP56001, for audio signal processing. External SRAM modules are used for program and data storage. A codec chip is employed to handle the digital interfacing between the DSP processor and the analog audio world.**

I. INTRODUCTION

Nowadays, personal computer (PC) has been getting more prevalent than ever with increasing power and, more importantly, decreasing cost. As such, its applications have been increasingly versatile, from the earlier word processing, to gaming, to wide-range software development, multimedia entertainment and many more. On the other hand, sound effects remained produced very much the same way by using stackable decks, custom made from electronic circuits. The audio processor card proposed in this paper is designed as an add-on card which plugs directly into the ISA bus of a PC to provide a more convenient means of creating and testing special sound effects for studio purposes.

The audio processor card is designed with a central processor that will execute an algorithm on a stream of audio signal inputs to produce certain special sound effects. Through an interfacing software at the PC front end, the user will be able to edit and assemble the programs used to generate sound effects and eventually download the assembled code to the processor card. The interfacing software will also allow the user to perform simple audio. This provides for future expandability and convenience. The processor would then run the program to create the sound effect.

A. Terminology

The definitions of some of the terms used in the remainder of this paper are given below.

Reverberation is a sound effect created within an enclosed area where audio signals are bounced back and forth with decreasing signal strength due to attenuation.
Chorus is a sound effect where the original audio signals are duplicated many times at different random times with random strength to collectively create many vocals even though there is only one.

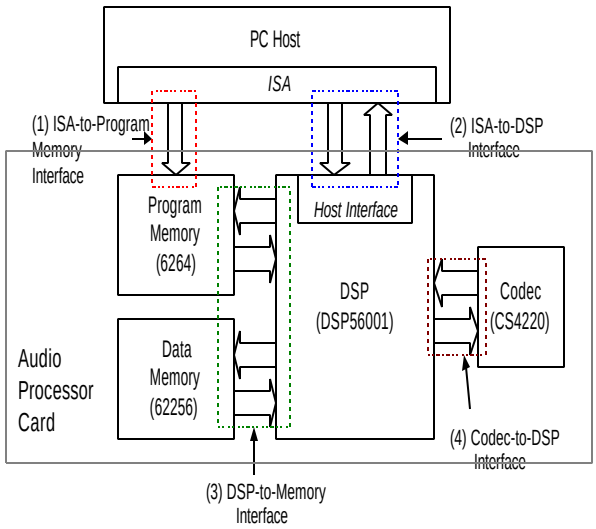

Fig. 1. System block diagram.

II. SYSTEM DESIGN

A block diagram of the overall system is shown in Fig. 1. The audio processor card is built around a Motorola DSP processor (DSP56001). The DSP56001 is a low-cost, high-performance 24-bit processor optimized for cost-effective consumer audio applications. External program memory (8K×24) and data memory (32K×24) are provided by interconnecting a number of high-speed SRAMs (6264 and 62256). A Crystal Audio Codec IC chip (CS4220) provides 24-bit analog-to-digital/digital-to-analog (ADC/DAC) conversion capability for the stereo audio signal.

The audio processor card is interfaced to the PC through the ISA bus. The program for the DSP56001 is downloaded from the PC to the external program memory chips directly. The PC also communicates with the DSP56001 processor during normal operation through the ISA bus. Besides, the DSP56001 fetches its program code from the external program memory and uses the external data memory to store the audio samples.

Signal input from an audio source (e.g. CD player or microphone) is fed to the CS4220 codec for digitization. The codec would then transmit the digitized signal to the DSP56001 using the interface module as shown in Fig. 1. The processed signals from the DSP56001 are then transmitted back to the codec for further digital-to-analog conversion before being output to a loud speaker or headphone.

A. DSP Processor

Fig. 2 shows the interfacing between the DSP56001 processor and the ISA bus. For this connection, the host interface (HI) of DSP56001 provides a convenient means for the PC to interact with the DSP56001 processor. The HI is a byte-wide, full-duplex, double-buffered, parallel port, which is connected directly to the data lines of the ISA bus. Address lines A14 and A15 together with IOW* and IOR* lines of the ISA bus provide the host enable input (HEN*) for the DSP56001 using an address decoder. A D flip-flop clocked by the OSC output from the ISA bus controls the R/W enable of the DSP56001 host interface.

B. Memory Module

Separate external program and data memory modules of size 8K×24 and 32K×24 respectively are used. Each memory module is organized as 3 separate modules, thereby making up the high byte, mid byte and low byte of the 24-bit word. The address and data lines from both the PC (ISA bus) and the DSP56001 feed the memory modules. The interface between the DSP56001 and memory modules is shown in Fig. 3.

The program code written in assembly language is housed inside the PC and the compiled code is downloaded from the PC to the audio processor card (external program memory)

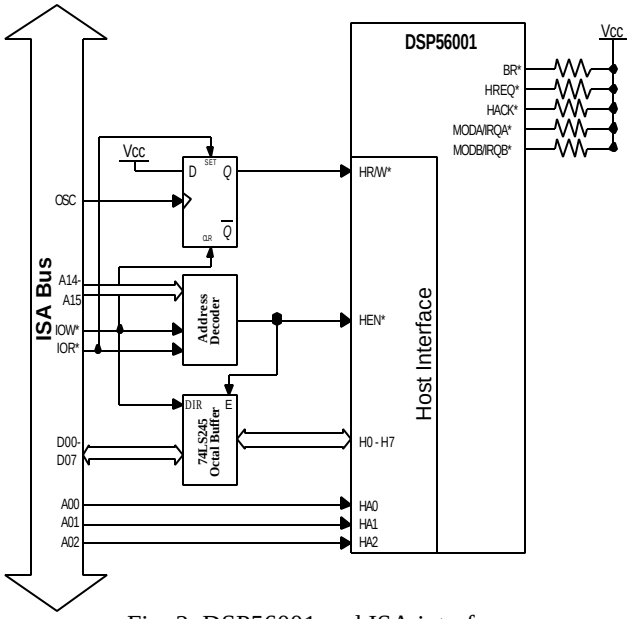

Fig. 2. DSP56001 and ISA interface.

through the ISA bus. The DSP56001 processor then executes the program stored in this external memory module.

To prevent bus contention, a control circuitry is included such that only either the program memory or the DSP is accessing the ISA bus at any one time.

C. Codec

A 24-bit Crystal stereo audio codec CS4220-KS is used to provide high quality analog-digital conversion. It basically consists of an ADC, a DAC, a low-pass filter and a sampler. The digitized audio samples from the codec are received and sent through the Synchronous Serial Interface (SSI) of the DSP56001 processor. The SSI consists of independent transmitter and receiver submodules and a clock generator. In addition, the transmitter and receiver are double buffered, which allow the data to be received and sent back at the maximum speed. These features provided by the DSP simplify the interfacing and programming for the codec.

The SSI uses four pins for interfacing - SSI Transmit Data (STD), SSI Receiver Data (SRD), SSI Serial Clock (SCK) and Serial Control 2 (SC2). These pins are wired to the Data Input (SDIN), Data Output (SDOUT), Serial Interface Clock (SCLK) and Left/Right Clock (LRCK) pins respectively of the codec through 500Ω resistors. In addition, the fundamental clock frequency, Master Clock (at pin XTI) is tapped from the oscillator clock of the DSP. The frequency used is 18.432MHz.

The codec is operated in its slave mode of operation. In other words, the DSP had to provide Left/Right clock (LRCK) and serial interface clock (SCLK) signals to the codec. The SCLK is used for transmitting and receiving audio data. Data will be valid on the rising edge of SCLK for both input and output. The LRCK is used to indicate left and right data and the start of a new sample period. The frequency of LRCK must be equal to the system sample rate, which is 48 kHz in this case. The 5V DC supply for the codec is tapped directly from ISA bus.

D. Software Modules

Three different software modules are written for the audio processor ISA card. They are:
a graphical user interface (GUI)
a device driver program (.DLL), and
a DSP program for creating special sound effects.

The GUI and the device driver programs reside in the PC and the DSP program is downloaded to the external program memory that is interfaced to the DSP processor.

1) *Graphical user interface*: The GUI program is designed and developed using Microsoft Visual Basic for its strength in rapid application development. The available functionalities built into the GUI program allow the user to edit, assemble a DSP source code, and download a particular DSP program to the card. The GUI also provides some basic

audio controls like volume, balance and left-right channels mixing.

In addition, inside the Sound Effect Param frame, two up-down controls are provided for the user to adjust up to two attributes for the sound effect. Each of the controls is sound effect specific, subjected to the inclusion of the right programming codes.

2) *.DLL driver program*: Underneath the GUI, the "dsp56kio.dll" driver program performs all the I/O tasks directly between the PC and the audio processor card via the ISA bus. Using the Microsoft Visual C++ software development tool, this program is written in C programming language for its strength in low-level I/O support.

The .DLL driver program handles the detailed implementation of the downloading of DSP programs from PC to the card via ISA bus. It also sends other control parameters (e.g. volume control) to the ISA card. As such, the GUI program would call the necessary functions from this driver program.

3) *DSP programs*: The DSP program contains the sound effect algorithm that will be executed by the DSP processor. The DSP programs are written in Motorola 56000 assembly language. These programs implement the sound effects through the DSP56001 processor. This program resides in the external high-speed SRAM chips. The DSP56001 processor would then execute the DSP program from these external program memory chips. The execution is carried out at the end of the downloading right after the DSP reset has been lifted. These memory chips are volatile and so would lose its contents when the PC is switched off.

III. Special Sound Effects

The algorithms for two of the sound effects implemented successfully - reverberation and the pitch - are presented here.

A. Reverberation

Reverberation (reverb for short) is probably one of the most heavily used sound effects in music. This is so because it resembles very much of the actual quality that we hear frequently. Sound waves from a particular source, say a speaker, take a direct path to reach our ears but can also take a slightly longer path by reflecting off a wall or the ceiling, before arriving at our ears with a weaker strength. This series of delayed and attenuated sound waves is what we call reverb, and this is what creates the 'spaciousness' of a room. Fig. 3 shows the impulse response of a typical reverberation. The implementation of the reverb could be illustrated as in Fig. 4.

Implementing the above algorithm using pure software means would require some buffers to store the previous set of input samples. Whenever a new sample is input, it would be added to the previous input sample depending on the delay. However, the delayed sample stored would need to be attenuated first by multiplying by a coefficient of less than 1 before adding.

B. Pitch Control

Fig. 5 below explains graphically what a pitch control does. In the frequency domain, pitch control refers to the shifting of the entire audio frequency band. As such, a vocal would sound like having a lower tone or the entire orchestra playing a music piece with a lower key. This frequency-shifting method produces the best results. However, its implementation requires the application of both Fourier transform and inverse Fourier transform. This translates into a huge requirement on processing power.

In its normal time domain, when an audio signal is output at a different rate than the input, a pitch change is actually attained as well. A higher output rate than the input creates a higher pitch and a lower output rate creates a lower pitch. This method is commonly used in many applications, e.g. Sonique Media Player for PC. The diagram in Fig. 6 illustrates this. However, when the signal is input in real-time, the output rate could not be different from the input rate. Thus a slight deviation from the second method is used – by using 'time stretching'. This is explained below.

If a circular buffer is divided into contiguous blocks of equal size, say 16 words of both left and right samples, each block could output at a longer time to its normal output sampling rate by outputting the first sample of each block once more. Refer to Fig. 7 for a diagrammatic explanation. When the block is extended by an extra sample and still outputting each sample in each and every block at an equivalent rate to the input sampling, a form of 'time-stretching' is attained. This time-stretching effect would be perceived to the human ears as a lower pitch.

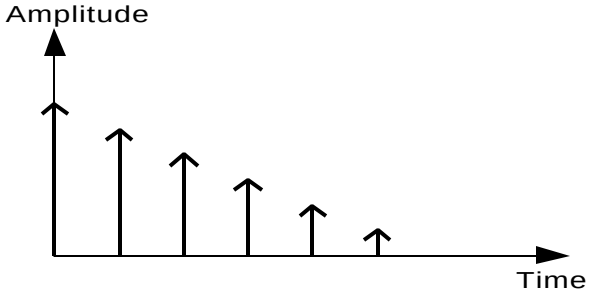

Fig. 3. Impulse response for reverberation.

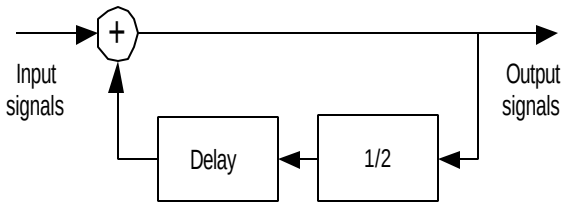

Fig. 4. Implementation of reverberation.

A similar method is used for higher pitch. Instead of outputting an extra sample to each block, a kind of 'time-compressing' could be attained by skipping a sample for each block.

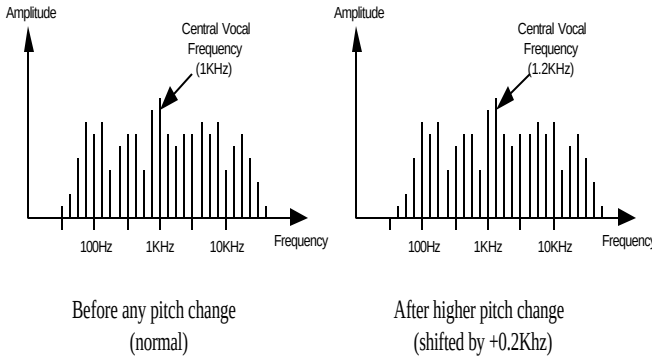

Fig. 5. Pitch control using frequency shifting.

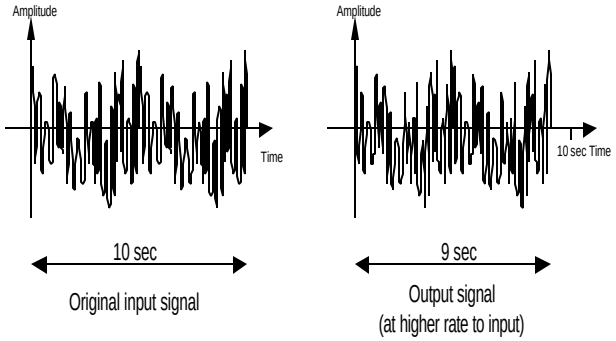

Fig. 6. Pitch control using higher output rate.

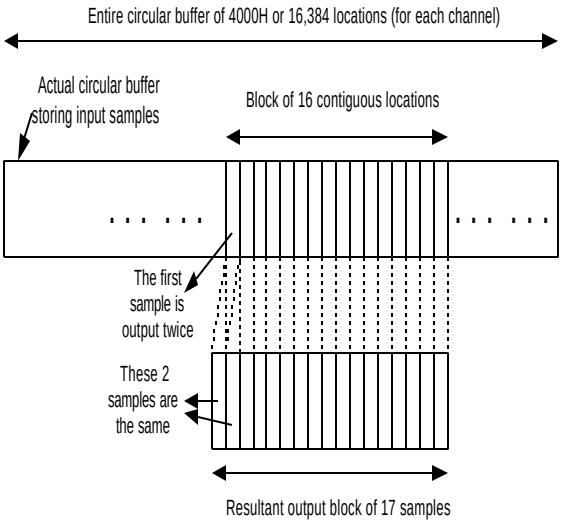

Fig. 7. Increased output samples for lower pitch.

IV. CONCLUSIONS

An audio processor ISA card for creating special sound effects is presented here. A convenient means of implementing and testing sound effects is achieved using this PC card that centers on a Motorola DSP56001 processor.

Two special sound effects are presented here. To accommodate larger programs or to introduce more working memory, more high-speed SRAM chips could be added onto the card. The central DSP chip could also be upgraded to DSP56002 for more processing power when the DSP program for the sound effect demands.

REFERENCES

[1] Motorola, Inc., *DSP56000/DSP56001 – Digital Signal Processor User's Manual*. Phoenix, AZ: Motorola, 1990.

[2] Motorola, Inc., *DSP56001A Semiconductor Technical Data*. Rev. 1, Phoenix, AZ: Motorola, 1997.

[3] B.B. Brey, *The Intel Microprocessors 8086/8088, 80186/80188, 80286, 80386 – Architecture, Programming, and Interfacing*. Upper Saddle River, NJ: Prentice Hall, 1997.

[4] Motorola, Inc., *Motorola Digital Signal Processing Development Software*. Phoenix, AZ: Motorola, 1990.

[5] K.C. Pohlmann, *Principles of Digital Audio,* 2nd ed., Indianapolis, IN: H.W. Sams, 1989.

[6] K. Steiglitz, *A DSP Primer: With Applications to Digital Audio and Computer Music*. Menlo Park, CA: Addison-Wesley, 1996.

Internet:

http://www.motorola-dsp.com

http://www.harmony-central.com/Effects/effects-explained.html