

FEATURE ARTICLE

Jerry Horn

A Comparison of Microcontrollers and DSPs

A Case Study

the more convinced I was that the answer really shows the differences between microcontrollers and DSPs, particularly when it comes to signal processing.

Let me first establish a baseline for this article. The gentleman wants to use either a Microchip PIC16F873 or Scenix SX28AC. I am familiar with both of these microcontrollers. He doesn't want to use a DSP. However, let me choose one for comparison. Because I am familiar with this DSP, I chose the Motorola DSP56364, a processor commonly used in many audio signal processing applications. Table 1 shows a comparison of the three devices.

Here are some of the features of each processor:

- PIC16F873—10-bit ADC with five input channels, two PWM, three timers, and USART
- SX28AC—"Virtual Peripherals" implemented in software (PWM, USART, timers, 8-bit ADC, etc.)
- DSP56364—Extended serial audio interface (ESAI), serial host interface (SHI), direct interface to DRAM/SRAM, 24 × 24 multiplier-accumulator, 56-bit barrel shifter, and PLL-based clocking

As you can see, the microcontrollers are inexpensive, small, and flexible. The DSP is larger, more expensive, and more specialized. Note that the only price I could find for the DSP is \$11 in low quantities. It might be less expensive in quantities of 100



I was recently given an opportunity to write about several of my favorite subjects—microcontrollers, digital signal processors (DSPs), and mixed-signal components, such as analog-to-digital converters (ADCs) and digital-to-analog converters (DACs).

This opportunity arose when a gentleman who is trying to make a guitar multi-effects unit was looking for information about how to implement digital audio effects using a microcontroller. He had difficulty finding any applications or algorithms for digital effects, so he turned to *Circuit Cellar's* ASK US section. The more I thought about his question,

Jerry gets all his favorites wrapped up in one, as an ASK US question about digital audio effects turns into a full-fledged article. What are the real differences between microcontrollers and DSPs with regards to signal processing? With this article, he leads us from delay to reverb, and it all started with one, not so simple, question.

Processor	Price (US \$)	Maximum MIPs	General-purpose I/O	Program I/O	Data RAM RAM	Package
PIC16F873	5.81 ⁽¹⁾	5 (20 MHz)	24	4096 × 14	192 × 8	28-pin
SX28AC	4.05 ⁽¹⁾	75 (75 MHz)	20	2048 × 12	136 × 8	28-pin
DSP56364	11.00 ⁽¹⁾	100 (100 MHz)	16	512–1280 × 24	1792–2560 × 24	100- or 112-pin

Note: ⁽¹⁾ is the price for a quantity of 100 pieces. ⁽²⁾ is the price for each quantity of one.

Table 1—Here you can see a variety of possible processors for a guitar multi-effects application.

(Motorola advertises \$4.95 for quantities of 250,000).

FINE TUNING

Now, let us turn our attention to the task to be solved. Many of the specifics in regards to his application were not available to me, so I had to speculate. For example, is this for his own personal use, or is he intending to develop a product? The answer to this question is important in order to resolve a number of issues.

I assume that he intends to make a fairly inexpensive product, but the volume of sales will not be high (e.g., total sales of several hundred units compared to a few thousand). This is a compromise between the two extremes of doing a product for yourself and designing something for high volume (e.g., 10,000 units or more during the life of the product).

The guitar effects that are provided by the device include delay, echo, chorus, reverberation (reverb), and distortion. I'll focus on the first four of these. As I discovered, distortion is a broad term that can mean just about anything. One type of distortion involves clipping the signal, either symmetrically (both top and bottom of

the signal) or asymmetrically (either top or bottom only). There are many other types of distortion, far too many to cover here. The first four items are enough for a general discussion.

Figure 1 shows the four types of signal processing to be accomplished. As you can see, items B through D are simply forms of item A, but with a gain element and a summation block.

So, all the effects can be accomplished with a variation of this algorithm: get the most recent ADC result, store it, retrieve an older result, multiply it by some gain value M (if needed), add the results together (if needed), and send the result to the DAC. Although it sounds like the multiplication by some gain value M is designed to increase the signal level, the intent is really to reduce its amplitude. Therefore, M is generally much smaller than 1. If allowed to be larger than 1, the maximum value may be 1.5 or 2 but not much larger. If M is greater than or equal to 1 in the case of reverb, positive feedback is present, and the ADC values keep getting larger and larger until digital saturation results and the values are all either positive full-scale or negative full-scale (which will not sound good at all).

For delay, the older ADC value can simply be retrieved and sent directly to the DAC (no math). For echo and chorus, older ADC values are combined with the most recent value, then the result is sent to the DAC. Changing the volume of the older samples will definitely be desired for echo and is likely to be needed for chorus as well. For reverb, older ADC values are retrieved, sent to the DAC, multiplied by a gain value, then combined with the most recent ADC value, and the sum is stored in memory. As mentioned, the older values must be multiplied by a factor less than 1 in order for the reverb effect to be stable. Too much reverb (values less than but still close to 1) can also cause the signal to satu-

rate, so some care must be exercised here.

TURN UP THE VOLUME

Let's take one of these effects and look a little closer at the details. Assume that the chorus effect must provide two delayed signals, each with an adjustable volume, and combine these with the main guitar signal. Thus, the device will be able to provide a chorus that sounds like three distinct instruments.

In such a case, for every ADC sample, the processor must perform the following actions:

- get most recent ADC result (or sample), call it SI
- store it
- retrieve result from J samples ago, call it SJ
- multiply SJ by gain MJ to get SJM
- retrieve result from K samples ago, call it SK
- multiply SK by gain MK to get SKM
- add SI , SJM , and SKM
- send the result to the DAC
- be finished before next ADC result is produced

The critical question is, can you accomplish this task with the processors that have been chosen?

It is not a simple question to answer, and I still need a few more details. For example, the conversion rate of the ADC, the resolution of the ADC, and the range of the gain value M are all needed.

Without going into the gory details, let me jump right to some answers. At a minimum, the ADC conversion rate must be 32 kHz and provide a resolution of 16 bits. I don't believe that any lower conversion rate or resolution will provide adequate quality. As a side note, most commercial effects units use 44.1- or 48-kHz ADCs with resolutions of 20 bits or higher.

I think M should probably be at least four bits (providing 16 different volume levels). Six bits would be better (64 levels), but it should not be necessary for M to be larger than eight bits. Unfortunately, human hearing is

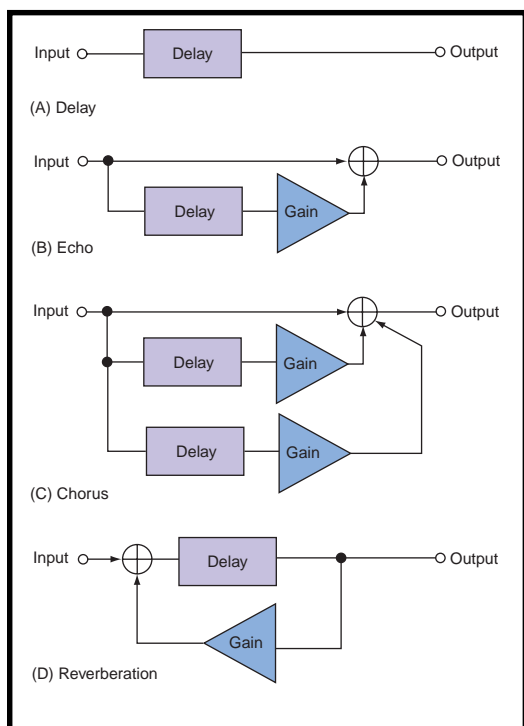


Figure 1—Here you can see the basic audio effects—delay, echo, chorus, and reverberation.

Value of M	Multiplier	Volume (dB relative to original)
255	0.996	-0
128	0.500	-6
64	0.250	-12
32	0.125	-18
16	0.063	-24
8	0.031	-30
4	0.016	-36
2	0.008	-42
1	0.004	-48
0	0.000	-Infinity

Table 2—Here you can see the values for M and volume levels.

not linear, and a simple approach to M is not adequate. For example, many volume controls change in increments of 1 or 0.5 dB. But, if M is changed from 1 to 2, then the signal level doubles, which is a 6-dB increment.

For simplicity, I will assume that M will be 8 bits and provides a possible range of 0 to 255. However, some settings may not be allowed. I will also limit M to provide gains of less than one. Table 2 shows the major values of M and associated volume levels.

This approach is not the best because, when M is lower than four, the volume increments are greater than 1 dB. However, the main intent here is to provide some numbers for discussion.

So, the processor must multiply a 16-bit number by an 8-bit number to produce a 16-bit result, and do this twice. It must also add three 16-bit numbers at the end. This must be accomplished in 31 μ s. This amount of time means each processor can perform the following number of instructions:

Processor	Number of instructions
PIC16F873	155
SX28AC	2325 at 75-MHz clock
SX28AC	1550 at 50-MHz clock
DSP56364	3100

There are many algorithms for multiplying two numbers within a microprocessor, which lacks a built-in multiplier. One algorithm involves a simple shift-add routine where, for each bit that is high in one of the numbers, the other is added to a running sum and then shifted in preparation for the next bit test. Another

involves a look-up table—a 4- x 4-bit look-up table would be appropriate for the 8-bit microcontrollers.

A rough calculation for a shift-add routine yields 240 instructions (or approximately 20 instructions per shift-add) for an 8 x 16 multiply. A look-up table method might cut this to less than 100 instructions, but not by much. So, the best that can be done is roughly 200 in-

structions for multiplying the two numbers in the chorus effect. Because there are many things that the microcontroller must accomplish (some of which have not even been discussed yet), I would immediately rule out the PIC16F873.

The SX28AC is still a contender. The chorus effect would eat up about 200 instructions for the two multipliers and some extra instructions for the additions at the end, call it 250 instructions. This means that about 16% of the microcontrollers' instructions are now being used, assuming a 50-MHz clock (11% if the clock rate is increased to 75 MHz). This is encouraging, but there are still many other items that the processor must handle.

For the DSP56364, there is no problem. It contains a 24 x 24 multiplier-accumulator that can multiply two 24-bit numbers in a single instruction. Adding some overhead for fetching the numbers and adding them together at the end still only results in less than 20 instructions. Here is where the real power of a DSP becomes apparent.

There is still an issue that has not been addressed. Exactly which ADC and DAC will be used, and how will they be operated? I have thought about this problem and there is no straightforward answer for the SX28AC. It is possible for the microprocessor to operate some standard ADCs and DACs, but there are several critical issues.

TIMING IS EVERYTHING

First, the sample-to-sample time of the ADC must be precise to within one instruction period. If the micro-

processor triggers the ADC to go into the hold mode, it must do so at a periodic rate that is exactly 31.250 μ s (for a 32-kHz conversion rate). If this signal is off by even one instruction from sample-to-sample, the signal-to-noise ratio will fall to 68 dB (with a 50-MHz clock and a 3-kHz full-scale input signal), or about 11 bits of performance. Although you can debate the actual performance relative to a "real-world" guitar signal, the main point remains that the conversion rate of the ADC must be precise.

It may be remotely possible to code the software within the SX28AC so carefully that an ADC, such as Burr-Brown's ADS8320, could be used for this application. Unfortunately this device is not intended for audio applications and is expensive compared to audio ADCs. And, the sample-to-sample period also applies to the latch signal supplied to the DAC.

A second issue with the ADC is the method of operation. In the case of the DAC, it is possible to clock out the digital word quickly and update the analog output by latching the DAC. However, the ADC must have a clock that allows each bit to settle during the conversion process. For example, the ADS8320 requires a clock with a 500-ns period. Fortunately, this clock does not have to be as precise (in time) as the sampling signal. So, if the conversion clock varies a few clock cycles, it is acceptable.

Burr-Brown has followed up the ADS8320 with some 16-bit ADCs that contain an internal clock, which removes the conversion clock issue. You simply get the most recent result from the ADC, trigger it for the next conversion, and move on. Unfortunately, these units are multi-channel, larger, and more expensive than the ADS8320. Other manufacturers offer similar devices, however, price remains the main issue along with the strict requirement for going into the hold mode (at the start of the conversion process) at a precise point in time relative to the last conversion.

Another option is to use external glue logic to drive the ADC, as well as to interface between the ADC and

the microcontroller (so that the ADC result is there when needed). The glue logic might also encompass the microcontroller to DAC interface. Still, this adds to the amount of money and design time needed. At a minimum, the requirement would be a number of standard 74x devices, although it might be preferable to use a CPLD or a small FPGA.

Using glue logic could also result in being able to use cheaper audio ADCs and DACs. Audio delta-sigma ADCs and DACs are inexpensive and high performance for the price, however, they must be operated by a low-jitter high-frequency clock. Universally, they are also serial devices, which support a limited set of serial interface standards, with the most popular being IIS (or I2S for Inter IC Sound). A CPLD would probably be required. A small FPGA might allow interfacing to an AC '97 codec, which provides a large number of features that could be useful in this application (at the very least, it is a cheap audio device providing a stereo ADC and a stereo DAC for under \$2).

STORAGE

There is one remaining issue that has not been discussed. All of the processors that have been included here cannot store a large amount of data, particularly the micro-controllers. So, how much storage is needed?

I searched for common delay times that are used in commercial guitar multi-effects processors but could not come up with a solid number. It appears that delays are in the 100-ms range, at the most. This appeals to my common sense because I don't think that longer delays would sound good. Still, I'll assume an absolute worst-case delay time of 250-ms, just to be conservative.

For 250 ms of storage, a 32-kHz conversion rate, and 16-bit ADC results, 16 KB of storage is needed. As it turns out, it is hard to find small SRAMs. The reason is that large SRAMs are inexpensive. Unfortunately they also require a large number of address and data pins. For 16 KB of memory, the microcontroller must provide a 14-bit address. If the SRAM

SX28AC + Non-audio ADC/DAC		SX28AC + Audio ADC/DAC		DSP56364 + Audio ADC/DAC	
Device	Cost (US\$)	Device	Cost (US\$)	Device	Cost (US\$)
SX28AC	4.05	SX28AC	4.05	DSP56364	11.00
128k x 8 SRAM	4.05	128k x 8 SRAM	4.05	128k x 8 SRAM	4.05
ADS8320ECT	14.10	PCM1800	3.25	PCM1800	3.25
PCM56	8.25	PCM1744	2.45	PCM1744	2.45
Glue Logic	?	Glue Logic	?	27C256	2.00
Total:	30.45 + glue	Total:	13.30 + glue	Total:	22.75

Table 3—Here you can see a basic bill of materials for the different possible designs.

has a parallel address port and an 8-bit parallel data port, then 22 I/O pins must be used, or glue logic must be added to the design.

I know that someone somewhere must make serial SRAMs with up to 16 KB of storage, but I could not find such a beast. Even with this, the SX28AC would have to bit-bang the serial interface, and this would require almost 800 instructions just to store one 16-bit number and retrieve two 16-bit numbers. So, a parallel device might be required even if a serial device is available.

None of these issues are of concern for the DSP56364. This device is designed to interface directly to serial audio delta-sigma ADCs through a variety of interface formats (I recommend the IIS format). In addition, it can directly interface to AC '97 codecs, which have a complex interfacing scheme. It also directly supports SRAMs as well as DRAMs. In fact, the interface to memory devices is clever and extremely flexible—yet another advantage of DSPs. (I am really not trying to sell DSPs here, but these are the facts for this application.)

PRICING

Table 3 shows a bill of materials for the three different system configurations. Prices generally consist of a single unit when purchased in lots of 100 (except for the DSP56364, for which I could only find a single unit price). The SRAM is overkill for this application, but all the configurations use it. Also, the DSP56364 works with only a single SRAM device even though it operates on 24-bit data internally. The DSP hardware can be programmed to save data as 8-, 16-, or 24-bit words, storing and retrieving

the data in successive addresses automatically (if larger than 8 bits).

Unlike the SX28AC, which can be programmed by the end user, the DSP56364 requires an external EPROM to store its program (in quantities of approximately 10k units or more, the DSP56364 can be manufactured with a specific program in mask ROM, removing the need for the EPROM—I am assuming that is not an option).

Of course, the unit would also require pedals, knobs, sliders, a case, jacks, and so on. However, these should be approximately the same regardless of the design approach to the effects circuitry. One difference would be power consumption. It seems like the DSP would require more power. Although it is possible that the DSP might be able to enter a sleep mode for a significant amount of time, it is unlikely that the SX28AC would be able to sleep.

A Scenix microcontroller, along with an audio ADC and DAC, would provide the lowest cost, but the savings might only be minimal. I do not honestly believe that the Scenix microcontroller can perform the necessary functions for all of the guitar effects unless they are severely limited in functionality. It is possible to get a useable system, but I would be concerned about the amount of work required to put 10 lbs of stuff in a 5-lb bag. You would have to be creative, careful, and meticulous in solving a large number of problems.

It is also my opinion that the DSP will make the whole design simpler and straightforward, leaving considerably more time to play with the actual effects and allowing different types of effects to be implemented, which is the real goal here.

I have used the Motorola DSP56x and DSP563x family of DSPs for almost 10 years. Do not let the number of pins fool you, the circuitry is straightforward, and these devices are more like microcontrollers on steroids than mysterious DSPs. They are not for every application, but where they are needed, they shine.

In addition, I typically program them just like microcontrollers. That is, the code is simple and does not use the signal processing hardware at all. This type of code probably makes up 95% to 99% of the program and is generally the user interface software and initialization code. The actual digital signal processing routines are a little more difficult to code, but those routines are usually small. And, after everything has been set up, you can play with the signal processing code, endlessly trying almost any type of signal processing you can think of.

ENCORE

And now, a few closing remarks. The DSP56362 is also a good candidate for this application. The onboard data RAM is 11k × 24 bits, which would provide up to 340 ms of storage at 32 kHz or 250 ms at 44.1 kHz. So, no external SRAM would be required. The DSP56362 is \$6 more than the DSP56364, but you save \$4 for the static RAM. And, the DSP56362 has more general-purpose I/O pins, as well as some additional hardware features (such as an S/PDIF output port). Combined with an EPROM and an AC '97 codec, the basic electronics cost is \$21 (compared to \$19 for the DSP56364 and an AC '97 codec).

There are a large number of DSP chips that could be used for this application. Texas Instruments, Analog Devices, and Motorola manufacture a variety of DSP chips. Cirrus Logic's Crystal Semiconductor division manufactures DSP chips that are mainly intended for audio signal processing, as does AKM Semiconductor.

A partial list of companies that manufacture audio ADCs, DACs, and codecs is available for downloading.

Finally, not all guitar effects require an ADC, DSP, and DAC. There is a thriving presence on the Internet

for classic guitar effects implemented strictly in analog. High-performance operational amplifiers (op-amps) are now inexpensive. So, an analog-only approach could be both cheap and high-performance. A list of a few Internet sites for old guitar effects circuits is available for downloading.



Jerry Horn is a senior design engineer with Lynium LLC, a consulting firm that specializes in testing and characterizing high-precision and high-speed mixed-signal components. His particular focus is on high-resolution and high-speed analog-to-digital converters, digital-to-analog converters, and high-performance audio converters. He is familiar with a wide variety of microcontrollers and digital signal processors, as well as the issues associated with interfacing these components to mixed-signal devices .

REFERENCES

- K.C. Pohlman, *Principles of Digital Audio*, Howard W. Sams & Company, Indianapolis, IN, 46268, 1989.
- J. Horn, "Clock Jitter and ADCs," <http://www.chipcenter.com/eexpert/jhorn/jhorn026.html>.

SOURCES

PIC16F873

Microchip Technology, Inc.

(888) 628-6247
(480) 786-7200
Fax: (480) 899-9210
www.microchip.com

SX28AC/Virtual Peripherals Scenix

(650) 210-1500
Fax: (650) 210-8715
www.scenix.com

DSP56364

Motorola, Inc.

(602) 952-4103
Fax: (602) 952-4067
www.mot-sps.com

ADS8320

Burr-Brown Corp.

(520) 746-1111
Fax: (520) 889-1510

www.burr-brown.com

DSP Chips

Texas Instruments, Inc.

(800) 477-8924, x4500
(972) 995-2011
Fax: (972) 995-4360
www.ti.com

Analog Devices, Inc.

(617) 329-4700
Fax: (617) 329-1241
www.analog.com

Cirrus Logic, Inc.

(510) 623-8300
Fax: (510) 226-2180
www.crystal.com

AKM Semiconductor, Inc.

(408) 436-8580
+(81) 6 6347 3133
Fax: (408) 436-7591
Fax: +(81) 6 6347 3132
www.akm.com

Circuit Cellar, the Magazine for Computer Applications.
Reprinted by permission. For subscription information,
call (860) 875-2199, subscribe@circuitcellar.com or
www.circuitcellar.com/subscribe.htm.