DEALING WITH A DIGITAL WORLD

Digital signal processors move to micros, where they handle complex data like sound and images with speed and flexibility

David A. Mindell

Last fall, Apple founder Steve Jobs introduced the NeXT machine, hailed as the first of a new generation of personal computers. It has a 17-inch "megapixel" display, a 256-megabyte magneto-optical disk drive, an Ethernet port, a SCSI port, and the multitasking Mach operating system, derived from Unix. While these features typify the growing sophistication of the personal computer marketplace, one chip in the NeXT "cube" represents a truly new direction in personal computer systems: a digital signal processor (DSP).

DSPs have been around for at least 10 years, mostly in high-end military or industrial applications and research. Recent developments in technology, however, have made single-chip DSPs widely available and affordable, as well as easily integrated into larger systems. Consequently, digital signal processing is expanding into numerous new applications, particularly on personal computers, and is one of the fastest-growing areas in digital technology today.

Telecommunications is a primary arena for digital signal processing, and many of today's high-speed modems use digital-signal-processing techniques to reduce errors and increase transmission rates. Specialized "adaptive" algorithms sense the noise and bandwidth properties of the telephone line and adjust transmission and filtering parameters accordingly. DSPs also encrypt and compress data for reasons of security and efficiency. Voice-mail systems, for example, process speech signals and convert them to compressed ASCII files for transmission over standard E-mail systems.

A single DSP chip can accomplish virtually all telecommunications functions within a system, eliminating the need for expensive additional hardware. The chip in the NeXT machine, for example, can act as a modem, a fax machine, a voice digitizer, and a data compressor while also serving as a high-throughput numeric processor.

The advent of digital audio has created a niche for DSPs in all areas of music processing, including synthesis, recording, mixing, equalization, and editing. "Direct to disk" has become very popular, where compact-disk-quality audio is recorded through a DSP in a personal computer onto a high-capacity hard disk drive. DSP systems can also clean up poor recordings, removing noise, and even replacing clicks and pops with interpolated artificial music. Some can perform time compression and expansion of speech and music signals without affecting the pitch. A 40-second message, for example, could fit into a 30-second commercial, all without "munchkinizing" or "Frankensteining." Despite its popularity, however, digital signal processing is still largely misunderstood by even the computer literati.

Digital Signals

The term *digital signal processing* refers to the digital implementation of filters and algorithms to process some kind of data or *signal*. These techniques, while more complex than their analog counterparts, provide all the advantages associated with numerical processing: speed, accuracy, increased noise immunity, greater dynamic range, flexibility and programmability, and the power to create sophisticated pseudo-intelligent systems.

Analog signals are *continuous-time* representations of a given quantity (see figure 1a). A microphone, for example, produces a varying voltage that is proportional to the sound it detects. The first problem of a digital system, then, is to convert these analog signals into numbers—that is, to digitize them. To do this, the system must *sample* at regular intervals to convert the continuous-time signal into a *discrete-time* representation (see figure 1b).

The time between these samples, the *sampling period*, is determined by Nyquist's sampling theorem, which states that samples must be taken at twice the highest frequency contained in the data. Audio signals, for example, have a 20-kHz bandwidth, that being the upper limit of human hearing. Digital audio, then, must take at least 40,000 samples per second to accurately reproduce sound (CDs actually operate at 44.1 kHz

continued

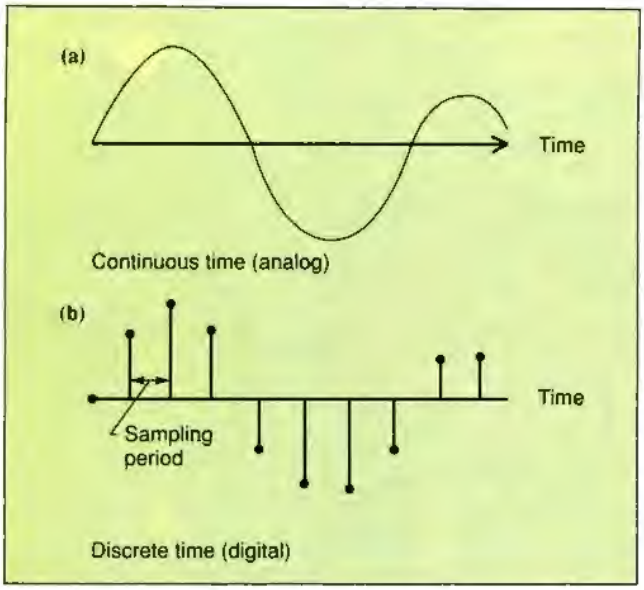

Figure 1: *In digitization,* **(a)** *an analog signal is sampled and converted to* **(b)** *a sequence of digital data. The data can then be passed through a digital signal processor for processing.*

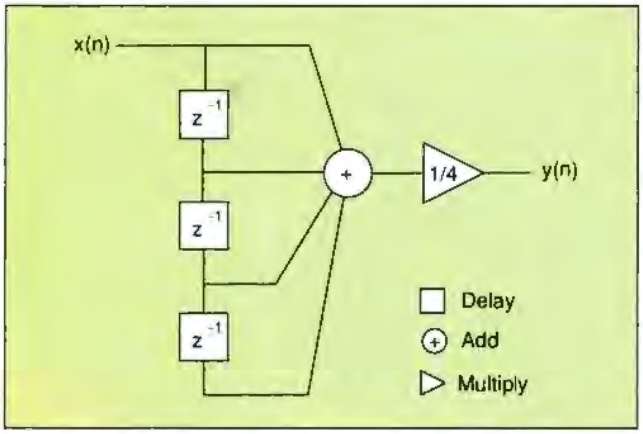

Figure 2: *In this diagram, a filter (a four-sample averager) adds an input* x(n) *to the three preceding inputs (which have been delayed for this purpose) and divides the total by 4 to give the output* y(n). *The effect of this averaging filter is to smooth out rapid deviations in the input signal, while leaving slower deviations, or low frequencies, relatively unaffected.*

to provide error correction and reduce noise).

An A/D converter converts a sampled analog voltage into a binary number. A DSP takes a string of numbers from an ADC, processes them in some way, and produces another string of numbers, which can then be passed through a D/A converter to reconstruct an analog signal. The digital data can also be passed directly to a computer for further processing or storage.

DSP Microprocessors

Until recently, most digital filters were hard-wired. A hardware multiplier was connected to an accumulator, which was connected to another multiplier, and so forth. Early microprocessors were simply not fast enough to perform the operations required for sophisticated filters. Advances in VLSI, however, have produced specialized DSP microprocessors with enough power to implement digital filters in software.

General-purpose microprocessors are bulky things. Loaded with features for memory management, system control, and compiler design, they can be clumsy to operate in real time (RISC architectures are an attempt to avoid this problem). DSP chips are similar to other microprocessors in that they execute programs, grab instructions and data from memory, and perform calculations. They are stripped down, however, and optimized for simple, repetitive operations with very high rates of data flow.

The distinguishing feature of DSP chips is their emphasis on the multiply-accumulate (MAC) operation, which is central to digital filtering. Current DSP chips—for example, the Texas Instruments TMS320C30, the Motorola DSP56001, and the AT&T DSP16A—can perform MAC operations in a single clock cycle in 60 to 80 nanoseconds, a value approximately equivalent to 25 to 33 million floating-point operations per second. (See the text box "A Look at DSP Chips" by John E. Hart on page 250 and table 1.) Impressive even by today's standards, such processing rates extend DSP chips' range well into the high-fidelity audio domain and just to the edge of video. Other features of DSP processors include extensive parallelism and pipelining, independent memories, and "bit-reversed" addressing modes for Fourier-transform data.

Another difference between general-purpose microprocessors and DSPs is that DSPs employ the Harvard architecture. In this scheme, data and instructions are kept in separate memories to allow the processor to perform several operations in parallel. There are numerous variations on this structure; some even allow access to five or six data banks simultaneously. The Motorola DSP56001, for example, has two data memories, denoted X and Y. For image processing, then, X and Y data can be kept separate, or for a complex Fourier transform, the X and Y memories can he used for real and imaginary data. A filtering operation might use the Y memory for the data stream and the X memory to hold the filter coefficients.

Most DSPs have some data and code memories on-chip and can access more memory through external buses. The on-board memories are small (rarely more than a few K bytes), but they are usually sufficient because DSP operations, while complex, produce relatively short programs. Some processors even include lookup tables for constants such as sine coefficients as part of on-chip ROM.

Because DSP chips are optimized for data throughput, there are usually several ways of presenting data to the CPU. Digital signals can enter through external buses, direct memory access, or one of several types of serial ports. These data paths, combined with flexible control features, also allow for several DSPs to be strung together to perform parallel operations.

The problem with DSP chips has been that, because they are so streamlined and were made with data flow and not systems in mind, they can be very difficult to program. Without convenient registers and instructions, compilers do not generate efficient code. Therefore, because high-bandwidth DSP applications often run under significant real-time constraints, critical routines must be written by hand and fine-tuned in assembly language. But the special architectures and extensive parallelism of most DSP chips mean that the assembly languages are obscure and esoteric and thus difficult to code.

Industry consensus is that the proliferation of DSPs has been slowed by these and other development difficulties. The situation is changing, however, as chip companies offer design aids such as emulators, library routines, and software simulators.

continued

A Look at DSP Chips

John E. Hart

The AT&T DSP32, introduced in 1985, was the first self-contained single-chip floating-point digital signal processor (DSP). Computer scientists and engineers found the compact architecture, ease of hardware integration, and impressive performance ideally suited to a variety of applications. It rapidly became clear that the DSP had great potential, in both single and multiprocessor configurations, to address a wide range of computational needs.

Although floating-point DSPs are only now beginning to appear in personal computers, competition between the major microprocessor designers has resulted in skyrocketing performance-to-cost ratios. Even though sample prices can be high, full-scale production DSPs are selling for as little as $40 per unit, or about $4 per million floating-point operations per second (MFLOPS). In what follows, I want to look at some of the more common floating-point DSPs, selected from table 1 on page 255.

The AT&T DSP32

The major AT&T DSP chips are the fixed-point DSP16 and DSP16A and the floating-point DSP32 and DSP32C. The DSP32 contains two processing units, the control arithmetic unit and the data arithmetic unit (DAU). The CAU is an integer processor with 16-bit resolution in the 25-MHz model; in the newer DSP32C, a 50-MHz device, it has 24-bit resolution. The DSP32 runs at 12.5 MFLOPS, while the DSP32C reaches a peak speed of 25 MFLOPS.

The CAU contains 21 registers that can be loaded from and stored to memory and that can be manipulated arithmetically in add, subtract, and various Boolean operations. Each operation takes four clock cycles. The first 14 CAU registers serve as address pointers to 32-bit floating-point operands held in either the internal or the external RAM; R_{15} through R_{19} can be used as post-index registers for floating-point memory accesses.

The DAU does floating-point computations exclusively. It has four 40-bit data accumulators (32-bit mantissas with 8-bit exponents), enabling "single-extended-precision" calculations within the DAU itself. All results are truncated or rounded to 32 bits when accumulators are stored to memory.

The generic DAU instruction involves two registers and three operands. It has the form

$$z = A_n = \pm A_m \pm y \times x$$

or

$$z = A_n = \pm y \pm A_m \times x$$

The A_j are the DAU accumulator registers ($j = 0$ to 3), and the variables z, x, and y can refer to specific address pointer registers within the CAU, to accumulator registers, or to an implicit 1 or 0. A DAU instruction statement occupies 32 bits of memory. The impressive floating-point speed of the DSP32 and its compact object codes is, in part, a result of the fact that one instruction can do *two* floating-point operations involving five variables.

The DSP32 is a four-state machine, with each instruction taking just four clock cycles. For example, if x, y, and z all refer to data in memory, using AT&T assembly language syntax, you could replace these variables with their address pointer registers $*R_n$, where $n = 1$ to 14. The instructions are then executed as follows:

Cycle 1: Fetch and decode the instruction.
Cycle 2: Fetch operand pointed to by the x-variable pointer $*R_n$.
Cycle 3: Fetch operand pointed to by the y-variable pointer $*R_i$.
Cycle 4: Do the floating-point operations and write the result to the z variable pointed to by $*R_j$.

There is a 32-bit memory access during each clock cycle. At the 40- to 50-MHz clock rates at which the DSP32s can run, this would require extraordinary memory performance. To make these high bandwidths possible, the memory is partitioned into two banks, Bank0 and Bank1.

The address location of the 4K bytes of internal memory is flexible. All the external memory, if included, is located in Bank0. Memory activity is interleaved between the banks to allow for two-cycle access to each bank. One bank is being addressed while the other is being accessed. Data flow on the *internal* 32-bit bus can proceed at a rate of one long word (or 4 bytes) per cycle, but on the external Bank0 bus, it proceeds at one long word for every two clock cycles. At 50 MHz, this interleaving yields an internal bus bandwidth of 200 megabytes per second.

A look at the four-state cycle sequence indicates that perfect implementation of this interleaving scheme requires a very careful allocation of data and instructions between the two banks. In practice, a useful programming compromise is simply to place data in one bank (usually the lower one, because this bank is expandable off-chip) and to put the instructions in Bank1.

The DSP32 connects to a host microprocessor through its 8-bit parallel interface (16-bit in the DSP32C). This interface features a cycle-stealing direct-memory-access (DMA) controller that allows the host to read or write to DSP memory without having to halt and restart the CAU or DAU processors via software. This ability to change data "on the fly" is central to uses of the DSP32 in interactive scientific teaching and research computing applications.

The serial I/O section of the DSP32 permits input and output of 8-, 16-, and 32-bit data. One use for this port is to connect DSPs together in multiprocessor systems. Another is to drive 16-bit D/A converters, providing a convenient high-speed analog data stream for monitoring computations in real time.

The Motorola 96002

The Motorola 96002's instruction set is a superset of that for the MC56000 (the fixed-point DSP), and the instruction mnemonics are similar to those for Motorola's general-purpose microprocessors. The floating-point chip will be available later this year in both a single-port (the 96001) and a two-port version (the 96002).

The 96002 chip features multiple internal and external buses, with internal memory arranged to support parallel transfers of program and operand data to and from the program controller and the data ALU, respectively.

Running at 26.6 MHz, the MC96002 can attain a peak throughput of 40 MFLOPS and 13.3 million instructions per second (MIPS), although the floating-point throughput will more typically be about 27 MFLOPS in 32-bit single precision or 43-bit single-extended precision. This high performance is a result of internal concurrency and parallelism. The program controller, address-generation unit (AGU), and data ALU operate in parallel. A typical instruction in the 96002 consists of a floating-point operation involving accumulator registers A_0 through A_9 in the ALU as sources and destinations, along with a parallel move.

While the ALU is executing a multiply-accumulate on several ALU accumulator registers, the AGU can be fetching two 32-bit numbers from each of two data memory banks and placing them in other data registers for use in a subsequent instruction. This latter data can be obtained from the internal RAM banks of *x*-data and *y*-data or from static RAM attached to the two external memory ports, A and B.

Both transfers use addresses contained in two of the eight pointer registers located in the AGU. The effective pointer addresses can be modified using index registers that are also contained in the AGU. At the same time that the floating-point operations and data transfers are occurring, the program controller prefetches and decodes the next instruction from the program memory. All this can occur in just one instruction cycle (two clock cycles).

The data ALU contains a single-cycle floating-point multiplier/accumulator that works with either 32-bit or 43-bit input data, the latter being made up of 32-bit mantissas and 11-bit exponents. The results are written to ALU registers in "infinite precision."

For example, a single-precision multiply produces a 48-bit mantissa. The result, stored in a 96-hit register, can be used in future register-to-register arithmetic operations without truncation. However, when a result is written from an ALU register to memory, it is automatically rounded down in hardware to single precision.

Double-precision calculations must be done in software, but the bus structure, in which the *x*-data and *y*-data can be concatenated, speeds up the transfer of double-precision data to and from the ALU registers. In addition, these 10 96-bit registers provide expanded capability for computing larger expressions than can be performed in the four DSP32 accumulators. To take one example, repeatedly used numeric constants can be permanently stored in some of these registers, avoiding the necessity of collecting them from memory each time they are needed.

The TI TMS320C30

The Texas Instruments TMS320C30 has several features in common with Motorola's 96002 and AT&T's DSP32. The CPU contains a floating-point multiplier and an accumulator, which operate on the eight 40-bit single-extended-precision accumulator registers, as well as on data directly transferred from memory. As in the DSP32, a multiply instruction can get its operands either from data registers (accumulators A_0 through A_7) or from memory locations pointed to by address pointers in the AGU. Like the 96002, the TMS320C30 has multiple internal and external buses.

The 320C30 uses a modified Harvard architecture. This means that there are separate data buses for instructions and data. Both program and data memories can be accessed at the same time via two address generators carried in the CPU unit. The internal zero-wait-state RAM is contained in two blocks of 1K byte by 32 bits. The on-chip memory also includes 4096 32-bit ROM locations and a 64- by 32-bit instruction cache.

The cache can be used for short but often-used subroutines, and the ROM can be used to hold code or constants that are common to a range of applications. Standard math libraries have been implemented in some ROMs (the DSP32, for one), and such ROM libraries save valuable memory space. The on-chip memory in all the DSPs occupies a substantial fraction of the chip's real estate. In the C30 chip, almost half the 700,000 transistors are related to memory.

The chip has four 24-bit address buses, a 24-bit peripheral bus, and three 32-bit data buses. The architecture facilitates rapid execution of operations involving two variables, such as dot products and correlations. The 320C30 is a two-state machine, and peak speeds, in which a multiply-accumulate is done in two clock cycles, reach 33 MFLOPS with the standard 60-nanosecond instruction cycle.

The TMS320C30 contains a large number of parallel arithmetic commands. A "normal" three-operand floating-point multiply instruction, specified by a 32-bit instruction, multiplies the contents of Source1 and Source2 and places the result in a destination register, which is one of A_0 through A_7.

In a parallel floating-point multiply-add instruction, the CPU takes operand data from four sources (registers or memory locations). The first two sources are multiplied together, and the second two are added. The results of these two operations, carried out in parallel in the same instruction cycle, are placed into two accumulator registers (which must be among A_0 through A_3). Two of the sources must be among accumulator registers A_0 through A_7, and the other two must be data from memory (accessed via reference to pointer registers in the AGU).

Parallel arithmetic can involve pairs of a wide range of arithmetic operations, including floating-point, integer, and bit-manipulation instructions. The indirect addressing modes include various indexing operations to facilitate rapid execution of vector algorithms.

The 320C30 has several peripheral interfaces, which should lead to easy integration of the chip into host systems. Two 8-megabit-per-second serial ports permit communication with other DSPs or external devices. There are two 32-bit parallel interfaces that can be attached to external memory, a 32-bit bus of a host CPU, or other processors in multiprocessor systems. With the on-board DMA controller, you can use the I/O ports concurrently without having to start and stop the CPU.

TI has just introduced a 29-MIPS, 16-bit DSP chip tagged the TMS320C5x. This new chip is an update of its TMS320 series of 16-bit fixed-point DSPs.

John E. Hart is a professor in the astrophysical, planetary, and atmospheric sciences department at the University of Colorado in Boulder. He can be reached on BIX c/o "editors."

Digital Filters

The basic signal-processing operation is filtering, which blocks or passes selected frequencies in the data. Filters come in several types: low-pass, which eliminates high frequencies; high-pass, which eliminates low frequencies; and band-pass and band-reject, which operate on specified frequency bands.

The simplest digital filter is an averager, also known as a *tapped delay line*. Consider an input stream $x(n)$ and an output stream $y(n)$, where n is the "index" of the digital samples.

The basic signal-processing operation is filtering, which blocks or passes selected frequencies in the data.

Then a "four-sample averager" can be constructed that implements the following equation:

$$y(n) = 1/4[x(n)+x(n-1)+x(n-2)+x(n-3)]$$

Thus, the output of the filter is the average of the present sample $x(n)$ and the three samples preceding it. Figure 2 shows this equation as a digital filter structure using standard notation, with adders (a circle with a plus sign), multipliers (a triangle with a gain value), and delays (boxes marked with a z^{-1}, indicating a delay by one sample).

The function of this filter is easy to understand: Rapid deviations in the input signal, or high frequencies, tend to get smoothed out by the averaging function. Slower deviations, or low frequencies, remain relatively unaffected. Thus, the four-sample averager implements a low-pass filter.

The logical extension of this basic filter is a discrete version of an operation called *convolution*. Convolution consists of taking a set of filter coefficients and "sweeping" them across the stream of input data (see figure 3). At each point, the output is determined by the sum of products of the coefficients of the input data:

$$y(n) = \sum_{k=0}^{f} x(n-k)b(k)$$

where $b(0)$ to $b(f)$ are the filter coefficients and $x(n)$ is the input data. The filter structure is then rewritten as in figure 3. Note the importance of the multiply-add operation, which, as I mentioned earlier, is reflected in DSP chip design. This basic filter is known as a nonrecursive or finite impulse response (FIR) filter; given an input (an impulse), its response will decay to zero when the input is removed. [Editor's note: *For more on convolution, see "Introduction to Image Processing Algorithms" by Benjamin M. Dawson, March 1987 BYTE, and "Finding the Titanic" by Marti Spalding and Ben Dawson, March 1986 BYTE.*]

The next level of complexity is a recursive filter with feedback; its output $y(n)$ depends not only on inputs $x(n)$ but also on

continued

Entering the World of DSPs

John E. Hart, Scott Kittelman, and Dan Ohlsen

While floating-point digital signal processors are showing up in top-of-the-line computer systems, you can already purchase DSP add-in cards for smaller systems. These cards allow you to implement many applications in areas such as chaotic dynamics, numerical analysis, and other compute-intensive scientific and engineering subjects. Table A shows a number of DSP add-in boards that are available for a variety of small systems, including the IBM PC and compatibles and the Macintosh.

If you'd like a little more hands-on experience, you may be interested in a project we developed at the University of Colorado as part of our research into the equations, formulated by E. N. Lorenz, that formed the basis for modern chaos theory. Our simple AT&T DSP32-based coprocessor board contains address-decoding logic, a 40-pin DSP32 with internal memory only, and two D/A output circuits that are driven from the DSP32's serial output. This coprocessor board can be attached to a PC, XT, AT, or 80386 machine that has the standard PC bus.

The board can be wire-wrapped on a PC prototype card, using documentation consisting of a layout diagram (for wire-wrapping), circuits, a parts list, and a wire-wrap list. To save cost, the board has no external memory, which would have to be expensive 30-nanosecond static RAM. However, the DSP32's 4K bytes of internal RAM is adequate for a wide range of small database problems, provided that I/O is handled externally through the host PC using the direct-memory-access capability of the DSP32 (e.g., many scientific problems that you would like to do interactively will be small enough to fit into the 1024-number internal capacity of the 40-pin DSP32). You can build a 25-MHz coprocessor board for about $250, or you can order it assembled and tested in a 30-MHz, 15-MFLOPS printed-circuit version called the FS-2, which requires an adjacent open slot for heatsink clearance (see below for details).

The software needed to operate the DSP32 must include some form of macro assembler that converts assembly language mnemonics into DSP32 machine code, and a device handler that can load programs and extract data from the DSP32 across the PC bus. A compiler that converts high-level language into DSP32 instructions is also helpful. The AT&T MS-DOS assembler-linker package costs $500, and its C compiler $1500.

Those who don't want to become involved in extensive low-level programming can obtain an inexpensive software package that includes a mini-BASIC compiler, a macro assembler, an interactive graphics-oriented controller, a FORTRAN interface, a small special-function library, and several demonstrations of the integration of ordinary differential equations and the generation of images using both BASIC and assembly language codes. This software can be used with either the assembled FS-2 board or your own wire-wrapped board.

John E. Hart is a professor and Scott Kittelman and Dan Ohlsen are research associates in the astrophysical, planetary, and atmospheric sciences department of the University of Colorado in Boulder. Plans for the DSP32 board are available for $5 from FASTec, Inc., 189 Mine Lane, Boulder, CO 80302, (800) 468-4142. (You can obtain the software package mentioned above for $95, including the manual, or the software package complete with the 15-MFLOPS FS-2 board for $399.95.)

Table A: *These companies make add-in digital-signal-processing boards for a variety of computer architectures, including ISA (the Industry Standard Architecture, on which the IBM PC AT and compatibles are built) and NuBus (Macintosh compatible). Development support (development system, assembler, C language, libraries, debugger, and simulator) is available for each board listed below. (Table courtesy of* DSP Update*)*

Company	Board	Bus	Processor	Width	Price
Ariel Corp. 433 River Rd. Highland Park, NJ 08904 (201) 249-2900	DSP-C25 PC-56	ISA ISA	TMS320C25 Motorola DSP56001	16-bit integer 24-bit Integer	$595 $595
Atlanta Signal Processors, Inc. 770 Spring St. Atlanta, GA 30308 (404) 892-7265	Banshee Chimera	ISA ISA	TMS320C30 TMS320C25	32-bit floating point 16-bit integer	$6995 $2195
Burr-Brown Corp. P.O. Box 11400 Tucson, AZ 85734 (602) 746-1111	SPV120 SPV125 ZPB32	VME bus VME bus ISA	TMS320020 TMS320C25 WEDSP32	16-bit integer 16-bit integer 32-bit floating point	$2995 $2995 $995
Communications Automation & Control, Inc. 1642 Union Blvd. Allentown, PA 18103 (215) 776-6669	DSP32-PC	ISA	AT&T DSP32	32-bit floating point	$1045
Data Cube, Inc. 4 Dearborn Rd. Peabody, MA 01960 (508) 535-6644	Euclid	VMEbus	ADSP-2100	16-bit integer	$5000
Digidesign, Inc. 1360 Willow Rd., Suite 101 Menlo Park, CA 94025 (415) 327-8811	Sound accelerator	NuBus	Motorola DSP56001	56-bit integer	$1295
Impact Technologies 2082-B Walsh Ave. Santa Clara, CA 95050 (408) 988-4980	Viper 8704	VME bus	Zoran VSP161	16-bit block floating point	$9950
Microstar Laboratories 2863 152nd Ave. NE Redmond, WA 98052 (206) 881-4286	DAP2400 series	ISA	Motorola DSP56001	56-bit integer	$2395-$3195
OKI Semiconductor 785 North Mary Ave. Sunnyvale, CA 94086 (408) 720-1900	PSP92	ISA	MSM6992	22-bit floating point	$6265
Sky Computer, Inc. Foot of John St. Lowell, MA 01852 (508) 454-6200	Challenge-S	P4 bus	TMS320020	16-bit integer	$5300
Spectral Innovations, Inc. 4633 Old Ironsides Dr., Suite 450 Santa Clara, CA 95054 (408) 727-1314	MacDSP series	NuBus	AT&T DSP32	32-bit floating point	$2295-$8995
Spectrum Signal Processing, Inc. 264 H St. Blaine, WA 98230 (604) 438-7266	56001 320C25	VME bus ISA	Motorola DSP56001 TMS320C25	24-bit integer 16-bit integer	$5995 $1995
Zoran Corp. 3450 Central Expy. Santa Clara, CA 95051 (408) 720-0444	VSPX series	ISA	ZR34161	16-bit integer	$1000-$3000

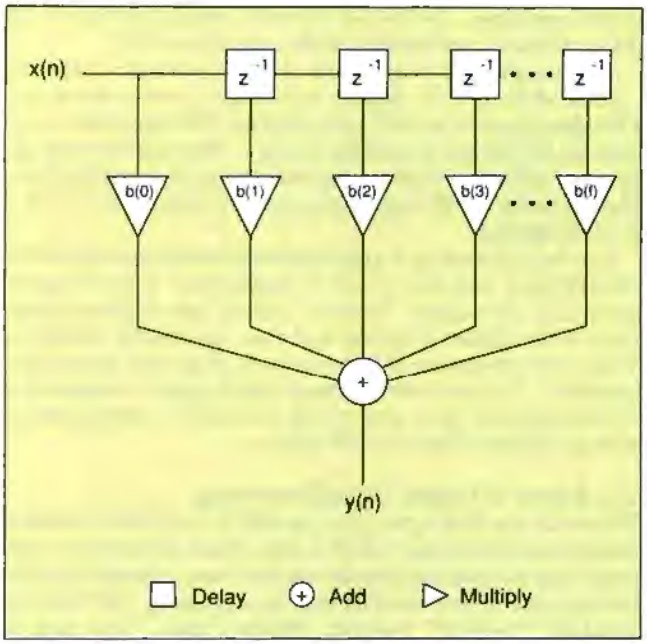

Figure 3: *The digital filter structure shown here performs discrete convolution. The output* y(n) *is the sum of an input* x(n) *and* k *previous inputs, each multiplied by a coefficient ranging from b(0) to b(f). Convolution can be used in a number of different applications, such as edge enhancement in image processing.*

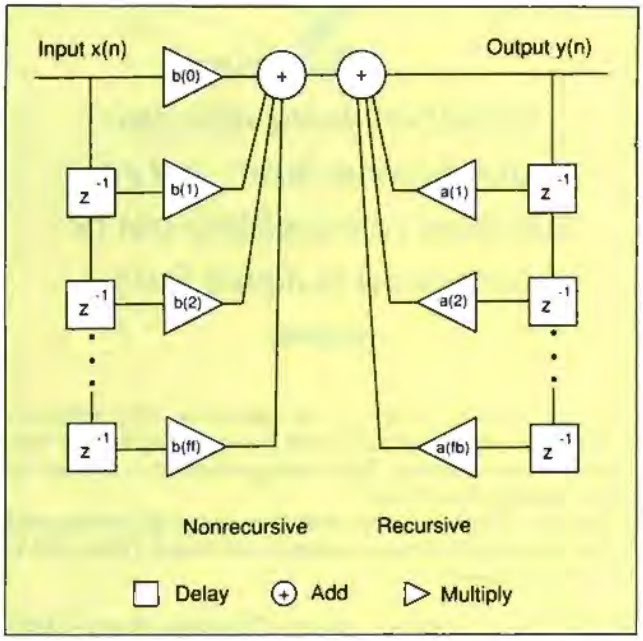

Figure 4: *This diagram of a general digital filter combines a nonrecursive section on the left (like those in figures 2 and 3) and a recursive section, in which the output* y(n) *is multiplied by a series of coefficients* a(k) *to* a(fb) *and added to the next output. Such filters are useful because their behavior closely models that of analog systems.*

Table 1: *Specifications for currently available digital-signal-processor chips (N/A = not available). (Information courtesy Nelson R. Manohar Alers and AT&T Bell Laboratories)*

DSP chip	Manufacturer	Year announced	Multiply operands	Multiply time	Technology design rule	Power dissipation (in watts)	Instruction cycle (in ns)	Data word length (in bits)
TMS32010	TI	1982	16×16 →32	200.0 ns	2.4μ NMOS	0.9	200	16
TMS320C25	TI	1986	16×16 →32	100.0 ns	1.8μ CMOS	0.6	100	16
TMS320C30	TI	1988	32×32 →E8	60.0 ns	1.0μ CMOS	1.0	60	32
DSP56001	Motorola	1986	24×24 →56	97.5 ns	1.5μ CMOS	N/A	97	24
DSP96001	Motorola	1988	24×24 →56	97.5 ns	HCMOS	N/A	75	32
DSP16	AT&T	1986	16×16 →32	55.0 ns	1.0μ CMOS	0.25	55	16
DSP16A	AT&T	1988	16×16 →32	25.0 ns	0.75μ CMOS	0.35	25	16
DSP32	AT&T	1985	32×32 →40	160.0 ns	1.5μ NMOS	2.0	160	32
DSP32C	AT&T	1988	32×32 →40	80.0 ns	0.75μ CMOS	0.8	80	32
μPD 7720SPI	NEC	1981	16×16 →31	250.0 ns	3.0μ NMOS	N/A	250	16
NEC 77230	NEC	1986	24E8 →47E8	150.0 ns	1.75μ CMOS	< 1.0	150	32
Intel 2920	Intel	1979	---- No multiplier ----		N/A	N/A	400	25
IBM RSP	IBM	1983	N/A	2 bits/cycle	2.0μ NMOS	2.5	200	16/24
HSP	Hagiwara	1983	12E4 →16E4	250.0 ns	3.0μ CMOS	0.25	250	20
ADSP2100	Analog Devices	1986	16×16 →32	125.0 ns	1.5μ CMOS	< 0.5	125	16
DSSP-VLSI	NTT	1986	12E6	N/A	1.2μ CMOS	0.7	50	18
MSM6992	OKI Electric	1986	16E6	100.0 ns	2.0μ CMOS	0.4	100	22
μSP32	Mitsubishi	1986	32×16 →32	150/450 ns	1.3μ CMOS	N/A	150	32
MB8764	Fujitsu	1986	N/A	N/A	N/A	0.3	100	16
TS68930	Thomson	1986	T16×16 →32	160.0 ns	N/A	N/A	160	16
NS LM32900	National Semiconductor	1986	16×16 →32	100.0 ns	2.0μ CMOS	0.5	100	16

Society's signals are being digitized: Letters become faxes, and even telephone conversations can be transmitted in digital form.

previous outputs $y(n-1)$, $y(n-2)$, and so on. This feedback, however, can cause ongoing or even diverging oscillations when input has been removed. Thus, these are known as infinite impulse response (IIR) filters.

Because filters can have both recursive and nonrecursive parts, a general difference equation for digital filters can be written as follows:

$$y(n) = \Sigma(k = 1 \text{ to } fb)\, y(n-k)a(k) + \Sigma(k = 0 \text{ to } ff)\, x(n-k)b(k)$$

where $a(k)$ is the feedback (recursive) coefficient and $b(k)$ is the feed-forward (nonrecursive) coefficient. Thus, the structure in figure 4 can be produced.

For an FIR filter, the recursive coefficients are set to zero. From this equation, you see that digital filters do nothing more than calculate a linear combination of current and previous inputs and previous outputs. The filter's frequency response depends on the function determining the coefficients for this combination.

FIR filters are easier to design than IIR filters because they are inherently stable. IIR filters must be designed to avoid unstable oscillations due to feedback. The signal delay in FIR filters is the same for all frequencies: a beneficial property called *linear phase response*. IIR filters, however, provide better response curves with fewer calculations.

Determining the best filter coefficients is a complicated task and involves selecting *poles* and *zeros* (solutions to the filter's characteristic equation, which determines its behavior) in the complex z-plane. CAD programs are now available that will produce optimized filters from specifications of frequency and phase response.

The Fourier Transform

Signals are composed of varying frequencies. A stereo system, for example, provides ways to control the frequency content of music. An increase in the treble control emphasizes the high frequencies. Increase the bass, and you'll hear the lows. Similarly, a prism breaks white light into its component frequencies, a process that reveals the spectrum's rainbow. Digital signal processing also has such a mechanism, a computational prism that analyzes signals in the frequency domain. It is called the Fourier transform.

The Fourier transform takes a signal in the time domain and converts it into the frequency domain, a process that reveals its spectrum. For digital signals where a continuous signal is represented as a set of points, the discrete Fourier transform is used. Because the DFT is computationally intensive, it has been optimized in the form of the fast Fourier transform. The FFT is a recursive routine that divides an initial signal into smaller and smaller pieces in order to perform 2-point DFTs as trivial operations. The results of these smaller operations are then scaled and combined to produce the entire FFT.

The straight DFT requires the order of n^2 complex multiplications, while the FFT requires only $n\log^2 n$, a reduction of over a hundred times for a 1024-point data set. FFT algorithms also have the advantage of working *in place*, meaning that they require no additional memory beyond storage of the initial data. Most of today's DSP chips can perform a 1024-point FFT in a few milliseconds.

I've been discussing a one-dimensional data model that fits chronological data like sound or temperature. Digital-signal-processing techniques, however, extend into higher dimensions. Convolution, filtering, and even the Fourier transform all have two-dimensional equivalents dealing with "spatial frequencies." For that reason, image processing is essentially a subset of digital signal processing, and today's image processors are often built around DSP chips.

The Future of Digital Signal Processing

Experts in the field agree that the DSP in the NeXT machine makes that system the first of a new breed of personal computer. Industry sources corroborate this view, reporting an imminent wave of new workstations incorporating DSP chips as standard "on-board" features. [Editor's note: *For a look at DSP boards currently available for personal computers, see the text box "Entering the World of DSPs" by John E. Hart et al. on page 252.*] Surely, as such systems proliferate and as DSP programming becomes simpler, there will be an explosion of diverse applications. Even more certain is that digital signal processing, like all truly innovative technologies, will extend beyond current visions and alter basic assumptions.

Society's signals are being digitized: Letters become faxes; records become CDs; speech is compressed and sent as mail. Even telephone conversations, paradigms of analog communications, are being transmitted through fiber-optic networks in digital form. But digital storage forces a kind of equivalence on various signals, removing them from their "real world" analog contexts.

In fact, digital signal processing is so broadly applicable only because, once inside a computer, "signals" are essentially all the same. Music, speech, codes, and even images can be converted to strings of numbers containing a given quantity of "information" to be distinguished and extracted from noise.

Recent controversies over digital audio tape are a good example of how digital techniques, with their capacity to make perfect copies, are calling into question concepts of originality and ownership in the information industry. Given the new equivalence it imposes on data, digital signal processing may require a rethinking of the very meaning of information: as a creation, as a signal, and as a commodity. ■

ACKNOWLEDGMENTS

The editors would like to thank Nelson R. Manohar Alers, member of the technical staff of AT&T Bell Laboratories, for providing information for table 1. Thanks also to DSP Update *(a monthly newsletter covering DSP markets, products, technology, and competition, published by BDG Publications, P.O. Box 3044, Stanford, CA 94309) for providing the information in table A, which accompanies the text box "Entering the World of DSPs" on page 252.*

David A. Mindell is a technical consultant with Exactitude Consulting based in Pittsford, New York. He also writes about how computers and digital processing are altering our conceptions of symbolic exchange. He can be reached on BIX c/o "editors."