Choosing a DSP Processor

Berkeley Design Technology, Inc.

Introduction

DSP processors are microprocessors designed to perform digital signal processing—the mathematical manipulation of digitally represented signals. Digital signal processing is one of the core technologies in rapidly growing application areas such as wireless communications, audio and video processing, and industrial control. Along with the rising popularity of DSP applications, the variety of DSP-capable processors has expanded greatly since the introduction of the first commercially successful DSP chips in the early 1980s. Market research firm Forward Concepts projects that sales of DSP processors will total U.S. $6.2 billion in 2000, a growth of 40 percent over 1999. With semiconductor manufacturers vying for bigger shares of this booming market, designers' choices will broaden even further in the next few years.

Today's DSP processors (or "DSPs") are sophisticated devices with impressive capabilities. In this paper, we introduce the features common to modern commercial DSP processors, explain some of the important differences among these devices, and focus on features that a system designer should examine to find the processor that best fits his or her application.

What is a DSP Processor?

Most DSP processors share some common basic features designed to support high-performance, repetitive, numerically intensive tasks.

The most often cited of these features is the ability to perform one or more multiply-accumulate operations (often called "MACs") in a single instruction cycle. The multiply-accumulate operation is useful in DSP algorithms that involve computing a vector dot product, such as digital filters, correlation, and Fourier transforms. To achieve a single-cycle MAC, DSP processors integrate multiply-accumulate hardware into the main data path of the processor, as shown in Figure 1. Some recent DSP processors provide two or more multiply-accumulate units, allowing multiply-accumulate operations to be performed in parallel. In addition, to allow a series of multiply-accumulate operations to proceed without the possibility of arithmetic overflow (the generation of numbers greater than the maximum value the processor's accumulator can hold), DSP processors generally provide extra "guard" bits in the accumulator. For example, the Motorola DSP processor family examined in Figure 1 offers eight guard bits.

A second feature shared by DSP processors is the ability to complete several accesses to memory in a single instruction cycle. This allows the processor to fetch an instruction while simultaneously fetching operands and/or storing the result of a previous instruction to memory. For example, in calculating the vector dot product for an FIR filter, most DSP processors are able to perform a MAC while simultaneously loading the data sample and coefficient for the next MAC. Such single-cycle multiple memory accesses are often subject to many restrictions. Typically, all but one of the memory locations accessed must reside on-chip, and multiple memory accesses can only take place with certain instructions. To support simultaneous access of multiple memory locations, DSP processors provide multiple on-chip buses, multi-ported on-chip memories, and in some cases multiple independent memory banks.

A third feature often used to speed arithmetic processing on DSP processors is one or more dedicated address generation units. Once the appropriate addressing registers have been configured, the address generation unit operates in the background (i.e., without using the main data path of the processor), forming the addresses

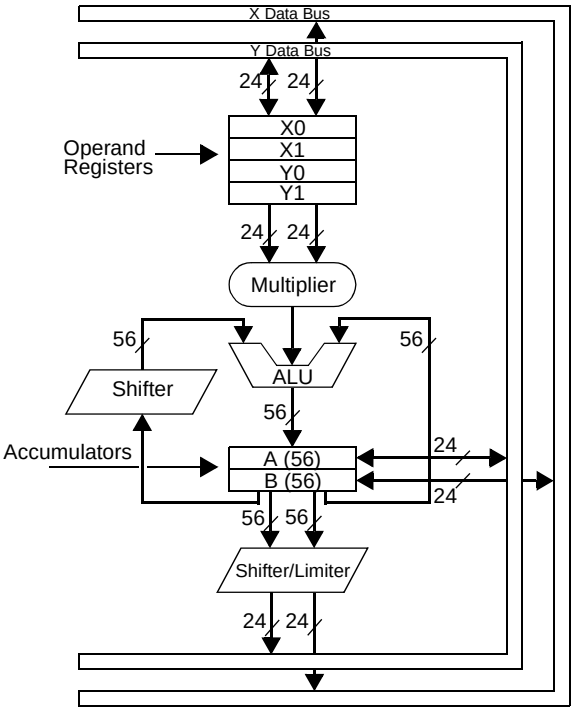

FIGURE 1. A representative conventional fixed-point DSP processor data path (from the Motorola DSP560xx, a 24-bit, fixed-point processor family).

required for operand accesses in parallel with the execution of arithmetic instructions. In contrast, general-purpose processors often require extra cycles to generate the addresses needed to load operands. DSP processor address generation units typically support a selection of addressing modes tailored to DSP applications. The most common of these is *register-indirect addressing with post-increment*, which is used in situations where a repetitive computation is performed on data stored sequentially in memory. *Modulo* addressing is often supported, to simplify the use of circular buffers. Some processors also support *bit-reversed* addressing, which increases the speed of certain fast Fourier transform (FFT) algorithms.

Because many DSP algorithms involve performing repetitive computations, most DSP processors provide special support for efficient looping. Often, a special *loop* or *repeat* instruction is provided, which allows the programmer to implement a *for-next* loop without expending any instruction cycles for updating and testing the loop counter or branching back to the top of the loop.

Finally, to allow low-cost, high-performance input and output, most DSP processors incorporate one or more serial or parallel I/O interfaces, and specialized I/O handling mechanisms such as low-overhead interrupts and direct memory access (DMA) to allow data transfers to proceed with little or no intervention from the rest of the processor.

The rising popularity of DSP functions such as speech coding and audio processing has led designers to consider implementing DSP on general-purpose processors such as desktop CPUs and microcontrollers. Nearly all general-purpose processor manufacturers have responded by adding signal processing capabilities to their chips. Examples include the MMX and SSE instruction set extensions to the Intel Pentium line, and the extensive DSP-oriented retrofit of Hitachi's SH-2 microcontroller to form the SH-DSP.

In some cases, system designers may prefer to use a general-purpose processor rather than a DSP processor. Although general-purpose processor architectures often require several instructions to perform operations that can be performed with just one DSP processor instruction, some general-purpose processors run at extremely fast clock speeds. If the designer needs to perform non-DSP processing, then using a general-purpose processor for both DSP and non-DSP processing could reduce the system parts count and lower costs versus using a separate DSP processor and general-purpose microprocessor. Furthermore, some popular general-purpose processors feature a tremendous selection of application development tools.

On the other hand, because general-purpose processor architectures generally lack features that simplify DSP programming, software development is sometimes more tedious than on DSP processors and can result in awkward code that's difficult to maintain. Moreover, if general-purpose processors are used only for signal processing, they are rarely cost-effective compared to DSP chips designed specifically for the task. Thus, at least in the short run, we believe that system designers will continue to use traditional DSP processors for the majority of DSP intensive applications. We focus on DSP processors in this paper.

Applications

DSP processors find use in an extremely diverse array of applications, from radar systems to consumer electronics. Naturally, no one processor can meet the needs of all or even most applications. Therefore, the first task for the designer selecting a DSP processor is to weigh the relative importance of performance, cost, integration, ease of development, power consumption, and other factors for the application at hand. Here we'll briefly touch on the needs of just a few classes of DSP applications.

In terms of dollar volume, the biggest applications for digital signal processors are inexpensive, high-volume embedded systems, such as cellular telephones, disk drives (where DSPs are used for servo control), and portable digital audio players. In these applications, cost and integration are paramount. For portable, battery-powered products, power consumption is also critical. Ease of development is usually less important; even though these applications typically involve the development of custom software to run on the DSP and custom hardware surrounding the DSP, the huge manufacturing volumes justify expending extra development effort.

A second important class of applications involves processing large volumes of data with complex algorithms for specialized needs. Examples include sonar and seismic exploration, where production volumes are lower, algorithms more demanding, and product designs larger and more complex. As a result, designers favor processors with maximum performance, good ease of use, and support for multiprocessor configurations. In some cases, rather than designing their own hardware and software from scratch, designers assemble such systems using off-the-shelf development boards, and ease their software development tasks by using existing function libraries as the basis of their application software.

Choosing the Right DSP Processor

As illustrated in the preceding section, the right DSP processor for a job depends heavily on the application. One processor may perform well for some applications, but be a poor choice for others. With this in mind, one can consider a number of features that vary from one DSP to another in selecting a processor. These features are discussed below.

Arithmetic Format

One of the most fundamental characteristics of a programmable digital signal processor is the type of native arithmetic used in the processor. Most DSPs use *fixed-point* arithmetic, where numbers are represented as integers or as fractions in a fixed range (usually -1.0 to +1.0). Other processors use *floating-point* arithmetic, where values are represented by a *mantissa* and an *exponent* as $\text{mantissa} \times 2^{\text{exponent}}$. The mantissa is generally a fraction in the range -1.0 to +1.0, while the exponent is an integer that represents the number of places that the binary point (analogous to the decimal point in a base 10 number) must be shifted left or right in order to obtain the value represented.

Floating-point arithmetic is a more flexible and general mechanism than fixed-point. With floating-point, system designers have access to wider *dynamic range* (the ratio between the largest and smallest numbers that can be represented). As a result, floating-point DSP processors are generally easier to program than their fixed-point cousins, but usually are also more expensive and have higher power consumption. The increased cost and power consumption result from the more complex circuitry required within the floating-point processor, which implies a larger silicon die. The ease-of-use advantage of floating-point processors is due to the fact that in many cases the programmer doesn't have to be concerned about dynamic range and precision. In contrast, on a fixed-point processor, programmers often must carefully scale signals at various stages of their programs to ensure adequate numeric precision with the limited dynamic range of the fixed-point processor.

Most high-volume, embedded applications use fixed-point processors because the priority is on low cost and, often, low power. Programmers and algorithm designers determine the dynamic range and precision needs of their application, either analytically or through simulation, and then add scaling operations into the code if necessary. For applications that have extremely demanding dynamic range and precision requirements, or where ease of development is more important than unit cost, floating-point processors have the advantage.

It's possible to perform general-purpose floating-point arithmetic on a fixed-point processor by using software routines that emulate the behavior of a floating-point device. However, such software routines are usually very expensive in terms of processor cycles. Consequently, general-purpose floating-point emulation is seldom used. A more efficient technique to boost the numeric range of fixed-point processors is *block floating-point*, wherein a group of numbers with different mantissas but a single, common exponent are processed as a block of data. Block floating-point is usually handled in software, although some processors have hardware features to assist in its implementation.

Data Width

All common floating-point DSPs use a 32-bit data word. For fixed-point DSPs, the most common data word size is 16 bits. Motorola's DSP563xx family uses a 24-bit data word, however, while Zoran's ZR3800x family uses a 20-bit data word. The size of the data word has a major impact on cost, because it strongly influences the size of the chip and the number of package pins required, as well as the size of external memory devices connected to the DSP. Therefore, designers try to use the chip with the smallest word size that their application can tolerate.

As with the choice between fixed- and floating-point chips, there is often a trade-off between word size and development complexity. For example, with a 16-bit fixed-point processor, a programmer can perform double-precision 32-bit arithmetic operations by stringing together an appropriate combination of instructions. (Of course, double-precision arithmetic is much slower than single-precision arithmetic.) If the bulk of an application can be handled with single-precision arithmetic, but the application needs more precision for a small section of the code, the selective use of double-precision arithmetic may make sense. If most of the application requires more precision, a processor with a larger data word size is likely to be a better choice.

Note that while most DSP processors use an instruction word size equal to their data word size, not all do. The Analog Devices ADSP-21xx family, for example, uses a 16-bit data word and a 24-bit instruction word.

Speed

A key measure of the suitability of a processor for a particular application is its execution speed. There are a number of ways to measure a processor's speed. Perhaps the most fundamental is the processor's instruction cycle time: the amount of time required to execute the fastest instruction on the processor. The reciprocal of the instruction cycle time divided by one million and multi-

plied by the number of instructions executed per cycle is the processor's peak instruction execution rate in millions of instructions per second, or MIPS.

A problem with comparing instruction execution times is that the amount of work accomplished by a single instruction varies widely from one processor to another. Some of the newest DSP processors use VLIW (very long instruction word) architectures, in which multiple instructions are issued and executed per cycle. These processors typically use very simple instructions that perform much less work than the instructions typical of conventional DSP processors. Hence, comparisons of MIPS ratings between VLIW processors and conventional DSP processors can be particularly misleading, because of fundamental differences in their instruction set styles. For an example contrasting work per instruction between Texas Instrument's VLIW TMS320C62xx and Motorola's conventional DSP563xx, see BDTI's white paper entitled *The BDTImark™: A Measure of DSP Execution Speed*, available at www.BDTI.com.

Even when comparing conventional DSP processors, however, MIPS ratings can be deceptive. Although the differences in instruction sets are less dramatic than those seen between conventional DSP processors and VLIW processors, they are still sufficient to make MIPS comparisons inaccurate measures of processor performance. For example, some DSPs feature barrel shifters that allow multi-bit data shifting (used to scale data) in just one instruction, while other DSPs require the data to be shifted with repeated one-bit shift instructions. Similarly, some DSPs allow parallel data moves (the simultaneous loading of operands while executing an instruction) that are unrelated to the ALU instruction being executed, but other DSPs only support parallel moves that are related to the operands of an ALU instruction. Some newer DSPs allow two MACs to be specified in a single instruction, which makes MIPS-based comparisons even more misleading.

One solution to these problems is to decide on a basic *operation* (instead of an instruction) and use it as a yardstick when comparing processors. A common operation is the MAC operation. Unfortunately, MAC execution times provide little information to differentiate between processors: on many DSPs a MAC operation executes in a single instruction cycle, and on these DSPs the MAC time is equal to the processor's instruction cycle time. And, as mentioned above, some DSPs may be able to do considerably more in a single MAC instruction than others. Additionally, MAC times don't reflect performance on other important types of operations, such as looping, that are present in virtually all applications.

A more general approach is to define a set of standard benchmarks and compare their execution speeds on different DSPs. These benchmarks may be simple algorithm "kernel" functions (such as FIR or IIR filters), or they might be entire applications or portions of applications (such as speech coders). Implementing these benchmarks in a consistent fashion across various DSPs and analyzing the results can be difficult. Our company, Berkeley Design Technology, Inc., pioneered the use of algorithm kernels to measure DSP processor performance with the BDTI Benchmarks™ included in our industry report, *Buyer's Guide to DSP Processors*. Several processors' execution time results on BDTI's FFT benchmark are shown in Figure 2.

Two final notes of caution on processor speed: First, be careful when comparing processor speeds quoted in terms of "millions of operations per second" (MOPS) or "millions of floating-point operations per second" (MFLOPS) figures, because different processor vendors have different ideas of what constitutes an "operation." For example, many floating-point processors are claimed to have a MFLOPS rating of twice their MIPS rating, because they are able to execute a floating-point multiply operation in parallel with a floating-point addition operation.

Second, use caution when comparing processor clock rates. A DSP's input clock may be the same frequency as the processor's instruction rate, or it may be two to four times higher than the instruction rate, depending on the processor. Additionally, many DSP chips now feature clock doublers or phase-locked loops

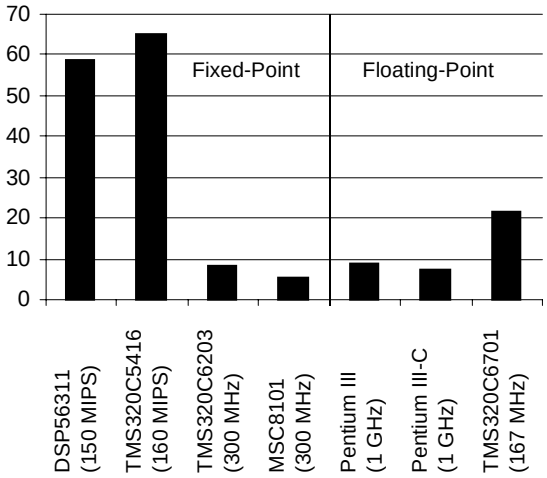

FIGURE 2. Execution times for a 256-point complex FFT, in microseconds (lower is better).

Note: Times are calculated for the fastest version of each processor projected to be available in June 2000. For the processors with on-chip cache, "-C" indicates performance with cache pre-loaded.

(PLLs) that allow the use of a lower-frequency external clock to generate the needed high-frequency clock on-chip.

Memory Organization

The organization of a processor's memory subsystem can have a large impact on its performance. As mentioned earlier, the MAC and other DSP operations are fundamental to many signal processing algorithms. Fast MAC execution requires fetching an instruction word and two data words from memory at an effective rate of once every instruction cycle. There are a variety of ways to achieve this, including multiported memories (to permit multiple memory accesses per instruction cycle), separate instruction and data memories (the "Harvard" architecture and its derivatives), and instruction caches (to allow instructions to be fetched from cache instead of from memory, thus freeing a memory access to be used to fetch data). Figures 3 and 4 show how the Harvard memory architecture differs from the "Von Neumann" architecture used by many microcontrollers.

Another concern is the size of the supported memory, both on- and off-chip. Most fixed-point DSPs are aimed at the embedded systems market, where memory needs tend to be small. As a result, these processors typically have small-to-medium on-chip memories (between 4K and 64K words), and small external data buses. In addition, most fixed-point DSPs feature address buses of 16 bits or less, limiting the amount of easily-accessible external memory.

Some floating-point chips provide relatively little (or no) on-chip memory, but feature large external data buses. For example, the Texas Instruments TMS320C30 provides 6K words of on-chip memory, one 24-bit external address bus, and one 13-bit external address bus. In contrast, the Analog Devices ADSP-21060 provides 4 Mbits of memory on-chip that can be divided between program and data memory in a variety of ways.

As with most DSP features, the best combination of memory organization, size, and number of external buses is heavily application-dependent.

Ease of Development

The degree to which ease of system development is a concern depends on the application. Engineers performing research or prototyping will probably require tools that make system development as simple as possible. On the other hand, a company developing a next-generation digital cellular telephone may be willing to suffer with poor development tools and an arduous development environment if the DSP chip selected shaves $5 off the cost of the end product. (Of course, this same company might reach a different conclusion if the poor development environment results in a three-month delay in getting their product to market!)

That said, items to consider when choosing a DSP are software tools (assemblers, linkers, simulators, debuggers, compilers, code libraries, and real-time operating systems), hardware tools (development boards and emu-

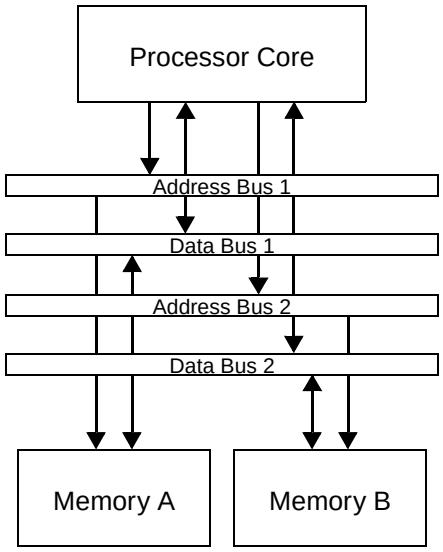

FIGURE 3. A Harvard architecture, common to many DSP processors. The processor can simultaneously access the two memory banks using two independent sets of buses, allowing operands to be loaded while fetching instructions.

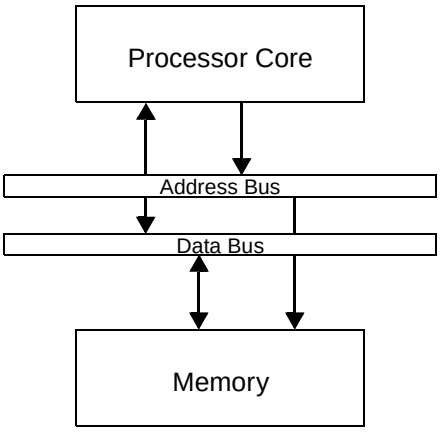

FIGURE 4. The Von Neumann memory architecture, common among microcontrollers. Since there is only one data bus, operands cannot be loaded while instructions are fetched, creating a bottleneck that slows the execution of DSP algorithms.

lators), and higher-level tools (such as block-diagram-based code-generation environments). A design flow using some of these tools is illustrated in Figure 5.

A fundamental question to ask when choosing a DSP is how the chip will be programmed. Typically, developers choose either assembly language, a high-level language—such as C or Ada—or a combination of both. Surprisingly, a large portion of DSP programming is still done in assembly language. Because DSP applications have voracious number-crunching requirements, programmers are often unable to use compilers, which often generate assembly code that executes slowly. Rather, programmers can be forced to hand-optimize assembly code to lower execution time and code size to acceptable levels. This is especially true in consumer applications, where cost constraints may prohibit upgrading to a higher-performance DSP processor or adding a second processor.

Users of high-level language compilers often find that the compilers work better for floating-point DSPs than for fixed-point DSPs, for several reasons. First, most high-level languages do not have native support for fractional arithmetic. Second, floating-point processors tend to feature more regular, less restrictive instruction sets than smaller, fixed-point processors, and are thus better compiler targets. Third, as mentioned, floating-point processors typically support larger memory spaces than fixed-point processors, and are thus better able to accommodate compiler-generated code, which tends to be larger than hand crafted assembly code.

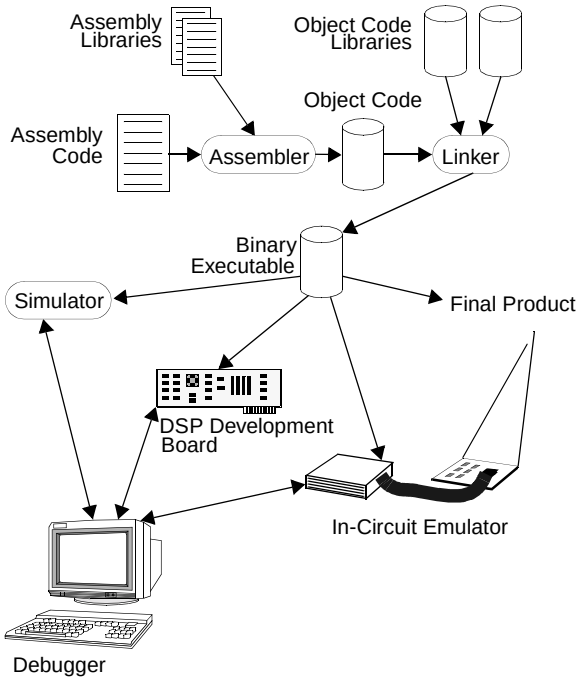

FIGURE 5. The interaction among assembly language development tools. These tools include assemblers, linkers, libraries, simulators, development hardware, and in-circuit emulators.

VLIW-based DSP processors, which typically use simple, orthogonal RISC-based instruction sets and have large register files, are somewhat better compiler targets than traditional DSP processors. However, even compilers for VLIW processors tend to generate code that is inefficient in comparison to hand-optimized assembly code. Hence, these processors, too, are often programmed in assembly language—at least to some degree.

Whether the processor is programmed in a high-level language or in assembly language, debugging and hardware emulation tools deserve close attention since, sadly, a great deal of time may be spent with them. Almost all manufacturers provide instruction set simulators, which can be a tremendous help in debugging programs before hardware is ready. If a high-level language is used, it is important to evaluate the capabilities of the high-level language debugger: will it run with the simulator and/or the hardware emulator? Is it a separate program from the assembly-level debugger that requires the user to learn another user interface?

Most DSP vendors provide hardware emulation tools for use with their processors. Modern processors usually feature on-chip debugging/emulation capabilities, often accessed through a serial interface that conforms to the IEEE 1149.1 JTAG standard for test access ports. This serial interface allows *scan-based emulation*—programmers can load breakpoints through the interface, and then scan the processor's internal registers to view and change the contents after the processor reaches a breakpoint. Scan-based emulation is especially useful because debugging may be accomplished without removing the processor from the target system. Other debugging methods, such as pod-based emulation, require replacing the processor with a special processor emulator pod.

Off-the-shelf DSP system development boards are available from a variety of manufacturers, and can be an important resource. Development boards can allow software to run in real-time before the final hardware is ready, and can thus provide an important productivity boost. Additionally, some low-production-volume systems may use development boards in the final product.

Multiprocessor Support

Certain computationally intensive applications with high data rates (e.g., radar and sonar) often demand multiple DSP processors. In such cases, ease of processor in-

terconnection (in terms of time to design interprocessor communications circuitry and the cost of linking processors) and interconnection performance (in terms of communications throughput, overhead, and latency) may be important factors. Some DSP families—notably the Analog Devices ADSP-2106x—provide special-purpose hardware to ease multiprocessor system design.

ADSP-2106x processors feature bidirectional data and address buses coupled with six bidirectional bus request lines. These allow up to six processors to be connected together via a common external bus with elegant bus arbitration. Moreover, a unique feature of the ADSP-2106x processor connected in this way is that each processor can access the internal memory of any other ADSP-2106x on the shared bus. Six four-bit parallel communication ports round out the ADSP-2106x's parallel processing features. Interestingly, Texas Instrument's newest floating-point processor, the VLIW-based TMS320C67xx, does not currently provide similar hardware support for multiprocessor designs, though it is possible that future family members will address this issue.

Power Consumption and Management

DSPs are increasingly being used in portable applications (such as cellular phones and portable audio players) where power consumption is a major concern. As a result, many processor vendors are reducing processor supply voltages and adding power management features to give programmers greater influence over processor power consumption. Power management features available on some DSPs include:

- **Reduced voltage operation.** Many vendors offer low-voltage (3.3-, 2.5-, or 1.8-volt) versions of their DSP processors. These processors consume far less power than five-volt equivalents at the same clock rate.
- **"Sleep" or "idle" modes.** Most DSPs feature modes that turn off the processor's clock to all but certain sections of the processor, reducing power consumption. In some cases, any unmasked interrupt will bring the processor back from sleep mode, while in other cases, only a few designated external interrupt lines will wake the processor. Some processors provide multiple sleep modes with different power savings and wakeup latencies.
- **Programmable clock dividers.** Some DSPs allow the processor's clock frequency to be varied under software control to use the minimum clock speed required for a particular task.
- **Peripheral control.** Some DSPs allow the programmer to disable peripherals that are not in use.

Regardless of power management features, it is often difficult for design engineers to obtain meaningful power consumption figures for DSPs. This is because a DSP's power consumption may vary by as much as a factor of three depending on the instructions it executes. Unfortunately, most vendors publish only "typical" or "maximum" power consumption numbers, usually without specifying what constitutes a "typical" program. One exception is Texas Instruments, which provides application notes that detail power consumption vs. instruction type and processor configuration.

Cost

Obviously, processor cost is a major concern for products that are to be produced in volume. For such applications, designers try to use the lowest cost DSP that meets the requirements of the application, even though such devices may be considerably less flexible and more difficult to program than costlier processors. Among processor families, the least expensive family members tend to have significantly fewer features, less on-chip memory, and lower performance than the more expensive members.

A key factor in processor pricing is the dependence of price on device packaging. For example, plastic thin quad flat pack (PQFP and TQFP) packages can be significantly less expensive than pin grid array (PGA) packages.

Finally, when considering prices, it is important to remember two things. First, processor prices are continually falling. Second, prices are strongly dependent on quantity, and prices for, say, a quantity 100,000 order may be significantly lower than for a quantity 1,000 order.

Summary

Despite some manufacturers' claims, there isn't a single best DSP chip. Rather, the right DSP depends on the application; a good choice for one application might be a poor choice for another. In this paper we have reviewed a number of criteria useful for choosing a DSP: arithmetic format, data width, speed, memory organization, ease of development, multiprocessor support, power consumption, and cost. Which of these are most important is a decision the system designer must make based on his or her application.

We conclude by mentioning two trends in DSP processor design. First, we expect to see more DSP proces-

sors tailored for specific high-volume applications, like cellular phones and portable digital audio players. The processors will integrate more system functions, such as analog-to-digital converters and LCD controllers, in an effort to reduce product cost and parts count. Second, we believe that more processors will be sold as licensable core designs, such as the Oak, Teak, and Palm cores offered by DSP Group, and the Carmel and TriCore cores offered by Infineon. As EDA (electronic design automation) tools advance in sophistication, system designers will find it easier to modify DSP cores and add custom peripherals to create highly specialized, cost-effective solutions for high-volume applications.

References

[1] *Buyer's Guide to DSP Processors*, Berkeley, California: Berkeley Design Technology, Inc., 1994, 1995, 1997, 1999. This 946-page technical report contains feature analyses and extensive benchmarking data for most popular DSP processors. The report provides profiling data from actual applications, such as modems and vocoders, to help designers assess the importance of specific processor features. Excerpts from this report, as well as a pocket guide to DSP processors, are available on the World Wide Web at www.BDTI.com.

[2] Phil Lapsley, Jeff Bier, Amit Shoham, and Edward A. Lee, *DSP Processor Fundamentals: Architectures and Features*, Berkeley, California: Berkeley Design Technology, Inc., 1996. An introductory textbook on DSP processor architectures which discusses how chip design affects performance.

[3] Will Strauss, *DSP Strategies 2002*, Tempe, Arizona: Forward Concepts, 1999. A 600-page report on the DSP chip market.

About Berkeley Design Technology

Berkeley Design Technology, Inc. (BDTI) is a software and technical services company focused on digital signal processing (DSP) technology. The company was founded by U.C. Berkeley faculty and researchers.

BDTI specializes in the analysis, benchmarking, evaluation, and development of technology used to implement DSP applications. Specifically, the company:

- Performs in-depth technical evaluations of microprocessors.
- Develops DSP application software and firmware.
- Publishes technical reports and books on DSP technology, including *Buyer's Guide to DSP Processors*, *Inside the Infineon Carmel*, and *DSP Processor Fundamentals*.
- Analyzes DSP algorithms and applications.
- Evaluates design tools and advises on tool selection and design methodologies.
- Develops specifications and recommendations for new DSP processors, software, and tools.
- Provides DSP-related training classes.

BERKELEY DESIGN TECHNOLOGY, INC.

2107 Dwight Way, Second Floor
Berkeley, CA 94704 USA
(510) 665-1600
Fax: (510) 665-1680
email: info@BDTI.com
http://www.BDTI.com

International Representatives

EUROPE
Cornelius Kellerhoff
Technology Products
Dusseldorf, Germany
+49 (211) 467 998
Fax: +49 (211) 467 999
niels@BDTI.com

JAPAN
Shinichi Hosoya
Japan Kyastem Co
Tokyo, Japan
+81 (425) 23 7176
Fax: +81 (425) 23 7178
bdt-info@kyastem.co.jp