The Digital Signal Processor Derby

The newest breeds trade off speed, energy consumption, and cost to vie for an ever bigger piece of the action

BY JENNIFER EYRE
Berkeley Design Technology Inc.

Applications that use digital signal-processing chips are flourishing, buoyed by increasing performance and falling prices. Concurrently, the market has expanded enormously, to an estimated US $6 billion in 2000. Vendors abound. Many newcomers have entered the market, while established companies compete for market share by creating ever more novel, efficient, and higher-performing architectures. The range of digital signal-processing (DSP) architectures available is unprecedented.

In addition to expanding competition among DSP processor vendors, a new threat is coming from general-purpose processors with DSP enhancements. So, DSP vendors have begun to adapt their architectures to stave off the outsiders.

What follows provides a framework for understanding the recent developments in DSP processor architectures, including the increasing interchange of architectural techniques between DSPs and general-purpose processors.

Performance through parallelism

Digital signal processors are key components of many consumer, communications, medical, and industrial products. Their specialized hardware and instructions make them efficient at executing the mathematical computations used in processing digital signals. For example, since DSP often involves repetitive multiplications, DSP processors have fast multiplier hardware, explicit multiply instructions, and multiple bus connections to memory to retrieve multiple data operands at once. General-purpose processors typically lack these specialized features and are not as efficient at executing DSP algorithms.

For any processor, the faster its clock rate or the greater the amount of work performed in each clock cycle, the faster it can complete DSP tasks. Higher levels of parallelism, meaning the ability to perform multiple operations at the same time, have a direct effect on a processor's speed, assuming that its clock rate does not decrease commensurately. The combination of more parallelism and faster clock speeds has increased the speed of DSP processors since their commercial introduction in the early 1980s. A high-end DSP processor available in 2000 from Texas Instruments Inc., Dallas, for example, is roughly 250 times as fast as the fastest processor the company offered in 1982. Even so, some of the newest DSP applications, such as third-generation wireless, are straining the capabilities of current DSP processors [see figure, p. 64].

As processors pick up speed, applications evolve to exploit all the extra horsepower—and then some. Thus, DSP processor designers continue to develop techniques for increasing parallelism and clock speeds.

How many instructions per clock cycle?

A key differentiator among processor architectures is how many instructions are issued and executed per clock cycle. The number of instructions executed in parallel, and the amount of work accomplished by each, directly affect the processor's level of parallelism, which in turn affects the processor's speed. Historically, DSP processors have issued only one instruction

per clock cycle, and have achieved parallelism by packing several operations into each instruction. A typical DSP processor instruction might perform a multiply-accumulate (MAC) operation, load two new data operands into registers, and increment the address pointers.

In contrast, high-performance general-purpose processors, such as the Intel Pentium and Motorola PowerPC, usually achieve parallelism by issuing and executing several fairly simple instructions per clock cycle. Why the difference?

DSP processors were originally designed for applications sensitive to cost, power consumption, and size. They relied on single-issue architectures, which are typically simpler to implement than architectures that issue more than one instruction per clock. Therefore, they consume less die area and power.

How wide an instruction?

Since the amount of memory a processor requires to store its software affects its cost, size, and power consumption, DSP processor instruction sets are designed with the goal of enabling highly compact programs. Thus, conventional DSP processors use a relatively short instruction word—typically 16 or 24 bits long—to encode each multi-operation instruction.

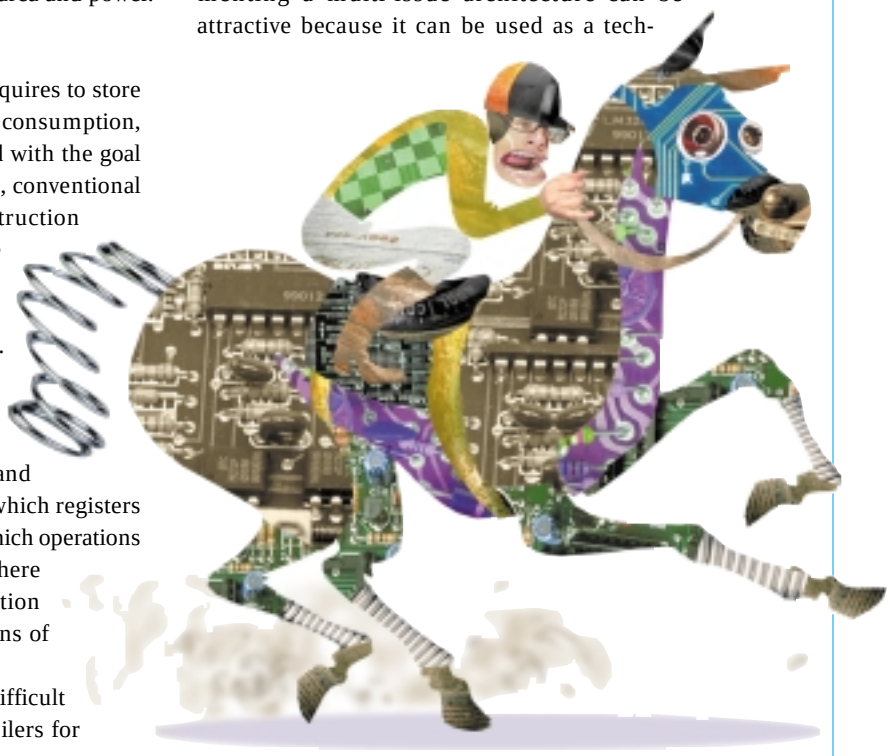
While this approach makes efficient use of program memory, it has several drawbacks. First, packing multiple parallel operations into a single, narrow instruction word means that the instruction sets tend to be highly constrained, filled with special cases and restrictions. There are often restrictions on which registers can be used with which operations, and on which operations can be combined in a single instruction. There are simply not enough bits in the instruction word to support many different combinations of registers and operations.

These complex instruction sets make it difficult to create efficient high-level language compilers for the processors. In reaction, most people who program conventional DSP processors (unlike those who program high-performance central processing units, or CPUs) sacrifice the portability and ease of programming offered by high-level languages and instead work in assembly language, that being the only way to harness the processor's capabilities effectively. This has become a significant disadvantage for conventional DSP processors, particularly as competition from more compiler-friendly general-purpose processors has grown.

An alternative to packing lots of operations in a single instruction is to use a technique common among general-purpose processors: include only one operation per instruction, and then issue a group of instructions in parallel. This is called a multi-issue approach. So, for example, the earlier multi-operation instruction might be split into five single-operation instructions: a MAC, two moves, and two address pointer updates. Each individual instruction is much simpler, but by executing them at the same time, the processor provides the same level of parallelism as does a single-issue processor that executes all of the operations as part of a single instruction.

Two advantages of this approach are increased speed and compilability, which come at the cost of added architectural complexity. Speeds increase because the use of simple instructions simplifies instruction decoding and execution, allowing multi-issue processors to execute at much higher clock rates (often two to three times higher) than single-issue DSP processors while maintaining similar (or higher) levels of parallel operation. The technique has given such general-purpose processors as the Pentium and the PowerPC clock speeds far higher than those seen on today's DSP processors.

This style of instruction set enables more efficient compilers, since compilers are able to better "understand" and use simple, single-operation instructions. Moreover, implementing a multi-issue architecture can be attractive because it can be used as a technique for upgrading the performance of an existing architecture while maintaining binary-level software compatibility. For example, the Pentium executes software written for the earlier 486 architecture, but issues and executes two instructions per cycle (where possible) rather than one. Accordingly, customers can upgrade the performance of their systems without having to recompile their software (a critical consideration for PC users, who typically lack access to the source code for their application software).



Multi-issue processors typically require more instructions (and hence more program memory) to implement a given section of software. Since each instruction is simpler than those of conventional DSP processors, more instructions are required to perform the same amount of work. In addition, multi-issue architectures often use wider instructions than conventional DSPs. The increased width (for instance, 32 bits rather than 16) primarily serves to avoid imposing limitations on, say, which registers can be used with which operations. (The wider the instruction word, the more distinct instructions

DAVEY LIU

can be specified, thus allowing the instruction set to support more combinations of operations and registers.) The removal of such limitations makes it much easier to create an efficient compiler for the processor.

However, the less memory used by a processor, the better, because memory affects its die size, cost, and power consumption. Hence, the fact that multi-issue architectures often require more program memory counts against them in many DSP applications.

That said, where performance is the ultimate driver, DSP architects have been willing to accept the penalties of a multi-issue approach in their DSP designs. Moreover, DSP processor architects realize that their customers are pushing harder for quality compilers, a longstanding weak point for DSPs. As DSP applications have grown from hundreds of lines of code to tens of thousands of lines, the benefits of programming in a high-level language have become a compelling factor driving DSP's migration to multi-issue architectures.

As previously mentioned, high-performance general-purpose processors often employ multi-issue architectures. However, there is a key difference in the way the multi-issue approach is implemented on these processors from the approach used on most multi-issue DSP processors.

Multi-issue DSPs typically use a type of architecture called very long instruction word (VLIW), the name for instructions that are grouped for parallel execution. Texas Instruments' VLIW-based TMS320C6xxx, for instance, can execute up to eight 32-bit instructions as part of a very long instruction word—so its VLIW width is 256 bits.

VLIW is one of two types of multi-issue architectures; the other is referred to as superscalar, and is the approach used in most multi-issue general-purpose processors. The two approaches differ mainly in how instructions are grouped for parallel execution. Current VLIW DSP architectures include the StarCore SC140 (from Agere Systems and Motorola); the Carmel core from Infineon, Munich; and the TigerSharc of Analog Devices Inc., Norwood, Mass. The only mainstream superscalar DSP processor currently available is the LSI40xx from LSI Logic Inc., Milpitas, Calif.

In a VLIW architecture, either the assembly-language programmer or the compiler must specify which instructions will be executed in parallel. In contrast, in a superscalar architecture, special hardware within the processor determines which instructions will be executed in parallel, based on the resources—such as registers—that are available, data dependencies as instructions are executed, and other considerations. This is determined when the program is executed. In other words, the superscalar processor shifts responsibility for instruction scheduling from the programmer or compiler to the processor.

A programmer's eye view

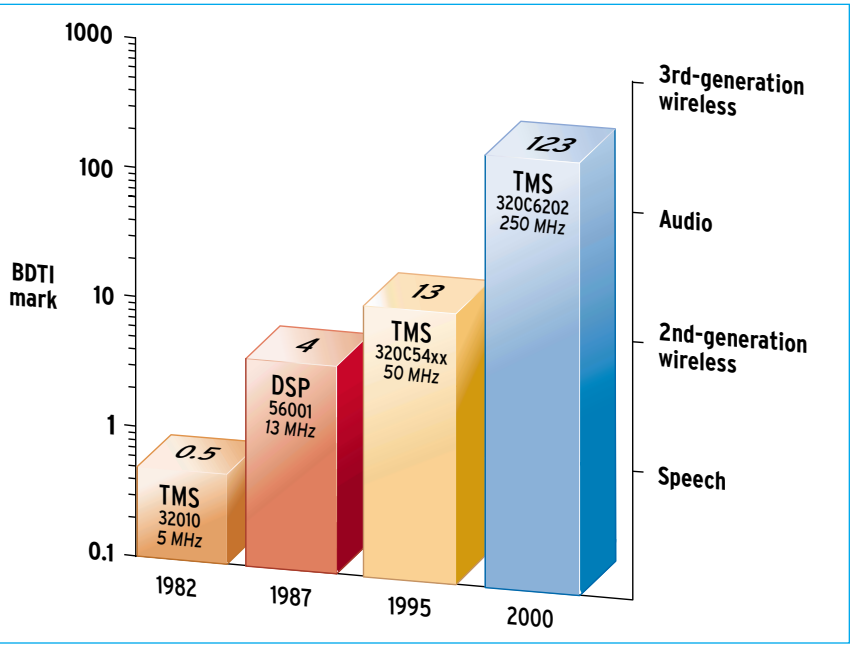
VLIW processors are often extremely tricky to program in assembly language, because the programmer must keep track of the multiple execution units on the chip and schedule multiple instructions for parallel execution. TI and other VLIW DSP processor vendors sidestep this issue by stating that today's advanced compiler designs can shoulder these burdens, allowing programmers to work in higher-level languages. However, it is still the case that compilers—even for VLIW processors—generally do not generate software that is as efficient or fast as that produced by skilled assembly programmers.

The superscalar approach makes possible binary compatibility between generations of processor architectures. In contrast, different generations of VLIW architectures will typically not be binary compatible, since information regarding instruction grouping is contained in the binary code. Thus, upgrading a VLIW-based processor to support more (or fewer) instructions to be executed in parallel would probably require software to be recompiled for the next-generation architecture.

Advocates of the superscalar approach say that it relieves the programmer or compiler developer from having to determine which instructions can be executed in parallel, thereby reducing

Four DSP Generations

Berkeley Design Technology Inc.'s composite digital signal processing (DSP) speed metric, the BDTImark, illustrates the changes in DSP processor speeds, set here against a backdrop of the speeds necessary for various existing and emerging DSP applications. A higher score indicates a faster processor. Note that the figure employs a logarithmic scale, as speeds have risen dramatically in the last 15 years.



programming complexity and enhancing compiler efficiency. All the same, achieving optimal performance on a superscalar processor often requires instructions to be arranged so that the processor will group them efficiently, in which case the programmer or the compiler is essentially responsible for instruction grouping after all.

A word on execution-time predictability

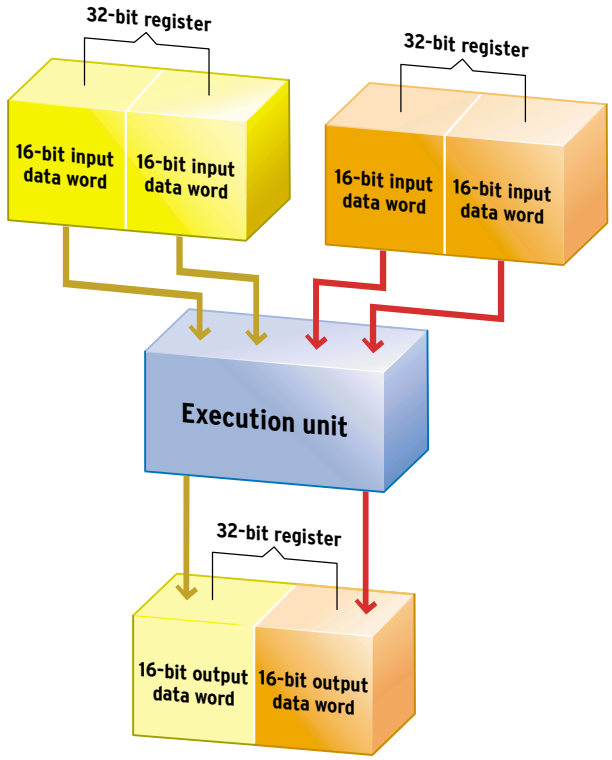
A serious drawback to the superscalar approach for DSP applications is that the programmer may not know exactly which instructions the processor will group together for parallel execution, and therefore may be unable to predict exactly how many clock cycles a given segment of code will run. The processor may group the same set of instructions differently at different times in the program's execution; for example, it may group instructions one way the first time it executes a loop, then group them some other way for subsequent iterations.

The lack of predictable execution timing can be a serious problem if the program must meet aggressive real-time processing deadlines. Of course, the programmer can assume worst-case timing and program accordingly, but then the processor's performance potential may remain untapped. (Note that this particular drawback is not an issue in processors used for PCs, since PC applications, like databases and word processors, typically do not have hard real-time constraints.)

Besides making real-time performance difficult to guarantee, execution-time unpredictability makes it difficult to optimize software. In DSP applications, which are computationally demanding amid stringent constraints on memory usage and energy consumption, software optimization is critical. A programmer who cannot easily predict the effect of software changes on program timing will find it difficult to assess whether an optimization is actually going to improve performance. In this case, software optimization becomes a trial-and-error process, and the result is likely to be far from optimal.

Processing Independent Data in Parallel

This example of single-instruction, multiple-data processing shows two 32-bit register values, each treated as two 16-bit pairs. An execution unit processes the pairs in parallel, producing two 16-bit results in half the time of a serial approach.



Improving on the concept

Although VLIW architectures tend to need more program memory than conventional DSPs, newer VLIW-based DSPs, such as the StarCore SC140 introduced in 1999, have found approaches to mitigate this difference.

To address the issue of high program memory usage, several VLIW-based architectures introduced in the last three years use mixed-width instruction sets (supporting, say, both 16-bit and 32-bit instructions). In this case, the processor uses short instructions when the full range of instruction options is not necessary, and reserves use of the longer instructions for performance-critical software sections that require the full power of the architecture. VLIW DSP processors employing this technique include Infineon's Carmel, the PalmDSPCore from DSP Group, Santa Clara, Calif., and the ST100 from STMicroelectronics, Geneva, Switzerland. StarCore's SC140 uses a very similar technique.

Last year, Texas Instruments created a new and improved version of its TMS320C62xx architecture, namely, the TMS320C64xx, which improves on the earlier processor's program memory usage. Where the 'C62xx uses extremely simple instructions, the 'C64xx includes several instructions that perform multiple operations—thus essentially reviving a technique used as long ago as the 1980s in conventional DSP processors. Fewer instructions are required for a given task, increasing code density, but making efficient compilers more difficult to develop. As in many areas of architectural design, processor architects continue to seek the best balance between conflicting objectives: in this case, low memory usage versus compilability.

Another approach to parallelism

Issuing multiple instructions per cycle is one way to increase a processor's parallelism. Parallelism can also be increased by using a single-instruction, multiple-data (SIMD) design. SIMD allows a processor to perform the same operation, using a single instruction, on multiple independent sets of data operands. Typically, a processor with SIMD support can treat data in long registers (for example, 64-bit registers) as multiple smaller data

words (say, four 16-bit words) on which it performs the same operation and generates multiple independent outputs.

SIMD became popular in the 1990s as a means of increasing existing CPU architectures' performance on the vector operations heavily used in multimedia and signal-processing applications [see figure, p. 65]. This approach dramatically increases the speed of processors on vector-oriented algorithms, where the operations are inherently parallelizable. The addition of SIMD has been a key enabler that has allowed general-purpose processors, such as the Intel Pentium III and Motorola PowerPC G4, to compete with DSP processors in terms of speed on DSP algorithms.

Although its roots are in general-purpose processors, the use of SIMD in DSP processors has spread and is now quite common among newer chips. But the level of support for SIMD operations varies widely.

A distinctly different approach to SIMD has been taken by Analog Devices. To create the ADSP-2116x, the company modified its basic conventional floating-point DSP architecture, the ADSP-2106x, by adding a second set of execution units, an exact duplicate of the original set. Each set of execution units in the ADSP-2116x includes a MAC unit, ALU, and shifter, and each has its own set of operand registers. The ADSP-2116x can issue a single instruction and execute it in parallel in both data paths using different data, effectively doubling its performance in some algorithms.

SIMD has its limitations, of course. Whether support for SIMD is strong or moderate, it is only useful if the data can be processed in parallel. For algorithms that are inherently serial—for example, those that use the result of one operation as an input to the next operation—SIMD is generally not of use. From the point of view of original-equipment manufacturers considering which DSP to buy, the key question they should ask is whether SIMD will be useful for their particular applications.

> For many DSP applications, speed is not the most important aspect of a processor's performance

From a programmer's perspective, effective use of a processor's SIMD capabilities can exact quite an effort. Programmers often must arrange data in memory so that SIMD processing can proceed at full speed, by, say, arranging data so that it can be retrieved in groups of four operands at a time. They may also have to reorganize algorithms to make maximum use of the processor's resources. The overhead required to arrange data or reorganize algorithms can lessen the improvement in performance due to SIMD, because the processor may have to execute additional instructions that rearrange data, load data in preparation for executing SIMD instructions, or sum all of the intermediate results.

Caches migrate into DSPs

Caches are small banks of fast memory located close to the processor core that are dynamically loaded with instructions and/or data. Their primary benefit is that they reduce the required speed—and hence the cost—of memory used by the processor. The clock speeds of many processors require extremely fast memory banks if the device is to operate at full speed. Use of a cache enables a processor to run at a high clock speed without requiring large banks of fast memory.

Caches serve to keep frequently used instructions or data close to the processor, acting as a holding tank between the processor and slower, larger banks of on-chip or off-chip memory.

A processor with a cache swaps needed instructions and data into the cache for quick accessibility, while executing other instructions. But if needed instructions or data are not in the cache, the processor waits while the cache is loaded with the required information. Thus, the amount of time required to execute a program will vary depending on the contents of the cache. As with superscalar architectures, caches introduce software execution-time variability.

Caches are common among general-purpose processors, but have been slow to catch on in DSPs because they add uncertainty to program execution time. But, recognizing the performance advantages that caches bring, processor architects are finding ways to adapt caches for their DSP applications.

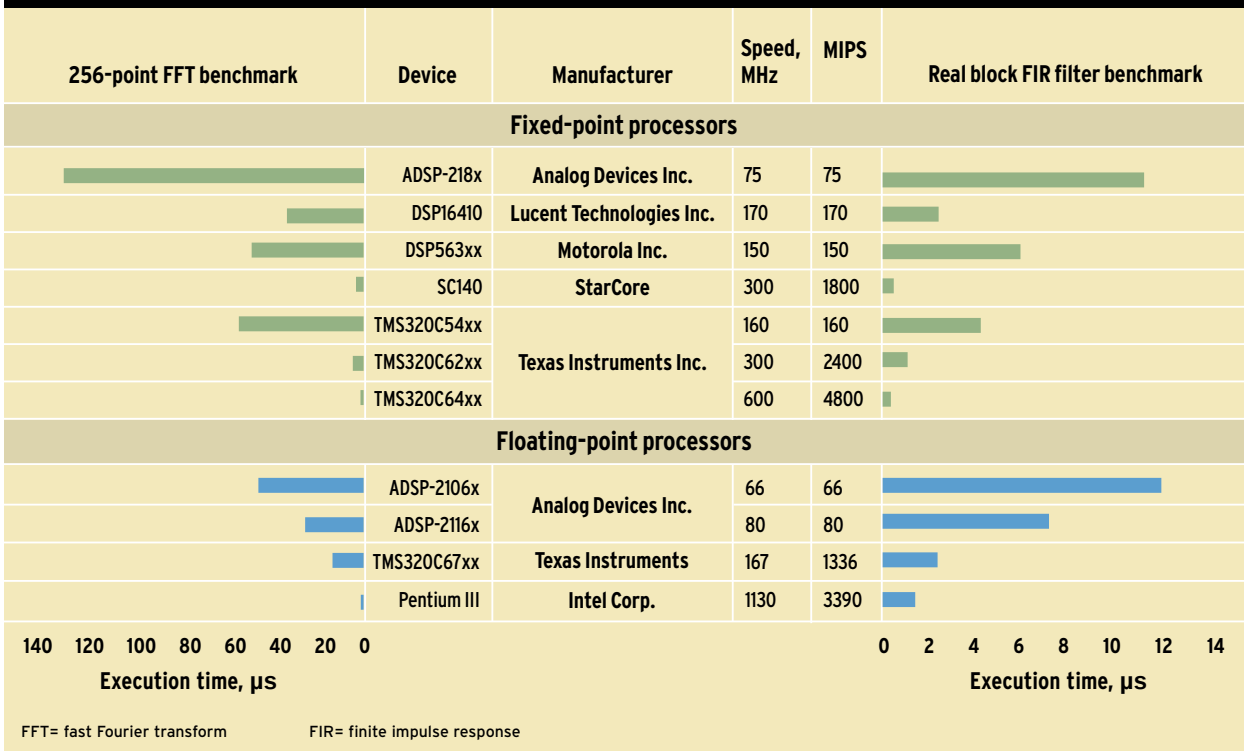
Until recently, the few DSPs that used caches had only instruction caches, not data caches, and these were simple enough that the programmer could still easily predict software execution times. Recently, however, a few new DSP architectures, such as Texas Instruments' 'C64xx, began using caches not unlike those that appear in general-purpose processors. This processor uses two-level caches on chip for instructions and data much like the two-level cache structure used in, say, the Pentium III. TI is presumably motivated by the same drive for higher clock speeds as the general-purpose processor vendors; samples of the TMS320C64xx, planned for this month, are expected to run at 600 MHz—twice as fast as the fastest mainstream DSPs currently available.

Caches present in DSP processors are typically adapted to suit DSP needs. For example, the processor may allow the programmer to manually "lock" portions of the cache contents, so that performance-critical sections of software can be guaranteed to be resident in the cache. This enables easy execution time predictions at the cost of reduced performance for other sections of software, which may have to be fetched from main memory.

If a DSP processor uses caches, it is essential that the vendor equip the programmer with tools that enable an accurate determination of program execution times. Thus far, the level of support in this regard has been disappointing. Texas Instruments, for one, does not currently provide the 'C64xx with an instruction-set simulator that accurately models the effects on execution times of cache misses (when the processor fails to find what it wants in the cache). For lack of such a tool, programmers will find it difficult to implement and optimize real-time DSP software, but have to make their best guess and use trial-and-error optimization.

Sample Benchmark Results of Leading Digital Signal Processors

Benchmark tests developed by Berkeley Design Technology Inc. measure the time it takes processors to run DSP algorithms. The DSP architectures here include conventional single-issue (Analog Devices' ADSP-218x and -2106x; Motorola DSP563xx; and Texas Instruments' TMS320C54xx); enhanced-conventional single-issue (the ADSP-2116x and Lucent DSP164xx); VLIW (StarCore's SC140, plus the TMS320C62xx, 'C64xx, and 'C67xx), and one high-performance superscalar general-purpose processor with SIMD enhancements (Intel's Pentium III). Also shown are their megahertz and millions-of-instruction-per-second ratings.



The bottom line: performance

From a user's point of view, it is not the architecture that matters—it is the architecture's effect on processor performance, in terms of speed, power consumption, and memory usage. Judging a processor's speed is seldom straightforward, however, since vendors use different (and often inconsistent) methods of evaluating DSP speed.

BDTI, an independent DSP analysis and software development firm in Berkeley, Calif., has defined its own set of DSP algorithm benchmarks, which include such common DSP algorithms as filters and fast Fourier transforms (FFTs). BDTI implements and optimizes the algorithms in assembly language on each target processor, reflecting current DSP programming practices. Benchmark implementations follow the functionality and implementation guidelines described in BDTI's benchmark specification document, the goal being to obtain apples-to-apples performance comparisons among processors [see chart above].

The processors that are listed in the chart include conventional and enhanced-conventional single-issue DSP architectures, VLIW DSP architectures, and one high-performance superscalar general-purpose processor with SIMD enhancements (the Intel Pentium III). For each processor, the megahertz and millions-of-instructions-per-second (MIPS) ratings are also shown.

A finding that surprises many people is that the DSP benchmark results for the Intel Pentium III at 1.13 GHz are faster than the results for all but the fastest DSP processors. This illustrates the point that DSP-enhanced general-purpose processors are providing increasing competition for DSP processors. However, speed is not the only metric of interest in this field; general-purpose processors are unlikely to replace DSPs any time soon because they typically are much more expensive, consume much more power, and do not have DSP-oriented development tools. They may also lack DSP-oriented peripherals and other integration, and (for superscalar general-purpose processors) they suffer from a lack of execution time predictability.

The future of DSPs

Over the past few years, many vendors have claimed to have produced the "world's fastest DSP." A key question engendered by these statements is: who cares? In comparing processor performance, there is a tendency to focus exclusively on speed. But for many DSP applications, speed is not the most important aspect of a processor's performance.

In fact, for the most commercially important DSP applications, obtaining adequate processing speed is not the major challenge. Rather, it is obtaining an appropriate level of processing speed while minimizing system cost, power consumption, required memory capacity, chip size, and application software and hardware development effort and risk. These factors often must be traded off against one another, and different applications place

Selected Mainstream DSP Processors

Vendor	Processor	Type of architecture	Year introduced	Notes
Analog Devices Inc. Norwood, Mass.	ADSP-21xx	Conventional	1986	Low-cost DSP; used in a variety of cost-sensitive applications, like modems
	ADSP-2106x	Conventional	1994	Floating-point DSP for military, audio, and imaging applications
	ADSP-2116x	Enhanced conventional	1998	Successor to '2106x—adds single-instruction, multiple-data (SIMD) capabilities
	TigerSharc	Very long instruction word (VLIW)	1998	Combines VLIW with extensive SIMD, targeting telecommunications infrastructure applications
LSI Logic Corp. Milpitas, Calif	LSI40xx	Superscalar	2000	The only commercial superscalar DSP processor
Motorola Semiconductor Products Sector Austin, Texas	DSP563xx	Conventional	1995	Low-cost DSP, often used in audio applications
Motorola Semiconductor Products Sector & Agere Systems Inc. Allentown, Pa.	StarCore SC140	VLIW	1999	High-performance DSP core developed jointly by Lucent Technologies and Motorola, each of which offers chips based on the SC140 core
Texas Instruments Inc. Dallas, Texas	TMS320C54xx	Conventional	1995	Low-cost, low-power DSP, mid-range speed, commonly used in cell phones
	TMS320C55xx	Limited VLIW	2000	Successor to the 'C54xx; VLIW, but executes only two instructions per cycle
	TMS320C62xx	VLIW	1997	First commercially successful VLIW DSP, used in telecommunications infrastructure and other high-performance applications
	TMS320C64xx	VLIW	2000	Successor to the 'C62xx—adds SIMD capabilities

different weights on each one. Achieving an appropriate balance for a wide range of applications is a key challenge facing processor vendors and users alike. Doing so depends only in part on processor architecture; many other factors, such as fabrication technology and development tools, are also critical.

General-purpose processors up the ante

As DSP-enabled general-purpose processors extend their reach, one might wonder whether DSPs will survive this stiff new competition.

Although DSP processors still typically have a strong advantage relative to general-purpose processors in terms of price and power consumption, this difference is not insurmountable for vendors of the latter. In addition, DSP processors have lagged behind their up-and-coming rivals in several areas unrelated to performance. Popular general-purpose processors tend to have better software development tools than DSP processors, particularly compilers. General-purpose processors, unlike most DSPs, are frequently available from multiple vendors.

In addition, general-purpose processor vendors have maintained software compatibility between generations, whereas historically, DSP processor vendors have not. Upgrading to a newer, faster DSP processor has typically required customers to learn a new architecture, new tools, and to rewrite their software completely—in assembly language, no less. These disadvantages of DSPs relative to general-purpose processors were not critical when the two classes were not in direct competition. But now that some general-purpose processors have the performance to compete with DSPs in DSP applications, these issues have become significant, and are influencing the direction of DSP processor development and business models.

DSP processor vendors have no intention of allowing their products to become obsolete. But they will have to adapt their designs to meet the general-purpose processors' challenge by addressing some of the weaknesses that have been, until recently, considered acceptable.

In addition, DSP vendors are now placing a heavier emphasis on compilers and other tools and on compatibility between generations. They are devising architectures that are more compiler-friendly by using multi-issue approaches and adding other architectural tweaks. They are boosting clock speeds by adopting caches and multi-issue architectures. In short, DSP processor vendors are scrambling to avoid losing their market share to the new breed of DSP-capable general-purpose processors. It will be interesting to see the new technologies that emerge from the scuffle. ●

Linda Geppert, *Editor*