

SOFTWARE TOOLS UNLEASH POWERFUL DSP SYSTEMS

This is the second part of a two-part report on trends in DSP software development tools. Part 1 appeared in the October 2000 issue.

With DSPs jumping in performance and spreading throughout the embedded world, software tools are heeding the call to speed development.

By Stan Runyon



The latest crop of DSP chips may be dazzling system developers with blazing speeds, but the real heat is being generated by advances in development systems and support software. Not only are the latest tools letting users wring out the best performance from the chips, but the tools are comfortable with the latest trends in system design: multifunctionality, multitasking, multiprocessing, dual DSP and general-purpose processor chips, and more.

For instance, using both a DSP chip and a general-purpose processor (GPP) makes development even more interesting, but it sparks engineers to look for software capabilities aimed at that

dual arrangement. Consequently, DSP tools are becoming easier to use with GPPs, and some development environments offer the ability to mesh DSPs and GPPs more efficiently. Even so, both the DSP chips and development and embedded software are growing so powerful that in some applications no GPP is needed at all; the DSP alone takes care of the system functions.

However, noting the muscle of the latest crop of chips, engineers at Delphi Communications Systems in Maynard, Mass., are “trying to remove the microcontroller from our soft radio designs for picocellular reference applications, leaving just a powerful DSP chip,” says Rick Kane, vice president of business development. One way to tap that power is to write applications that comply with the TMS320 DSP Algorithm Standard. Kane notes that Delphi has one of the largest such algorithm inventories.

“The standard’s rules impose a standard of excellence on all third parties,” Kane says, “and our customers benefit from consistency and assurance that all will fit together. For example, rules ensure compatibility with C register conventions, reentrancy, relocatability, documentation of memory type and usage, and so on. That way, our users can allocate resources intelligently.”

“The rules impose an object-oriented design technique, and that’s good from our point of view in terms of helping our customers with integration,” Kane adds. (For more information on the algorithm standard, see page 14.)

Until recently, a typical DSP executed a single task or function, most likely running code in one giant loop. No longer. Today, many systems are designed to be

multifunctional or multitasking, calling for new kinds of development capabilities—perhaps an embedded real-time kernel or operating system.

KERNELS LINE UP FOR EMBEDDING

An embedded kernel can be thought of as one wavelet in a gathering of forces that threatens to become a tsunami in software design—that is, the construction of operating system software and applications from an off-the-shelf library of software components as well as in-house code.

Besides a kernel, new compiler and configuration technologies are bearing down on some common objections to generalized commercial software components. Those objections are the lack of application specificity, the hit to performance, and bloated code.

Imagine, however, techniques that allow a software library to adapt itself, remove unnecessary code, configure itself to the specific requirements of customers’ applications—even provide for debugging of a commercial black box. That’s not a dream but, increasingly, a reality.

Embedded Internet audio applications, such as MP3 players, the Sony Music Clip, and other appliances (actually, anything that is power-, space-, or cost-constrained)

are ripe for such capabilities.

Typically, a miniature audio device can contain a DSP chip that runs the main MP3 decoding algorithm but also has enough power left over to perform other system functions: the user interface, communications or other functions usually relegated to a microproces-

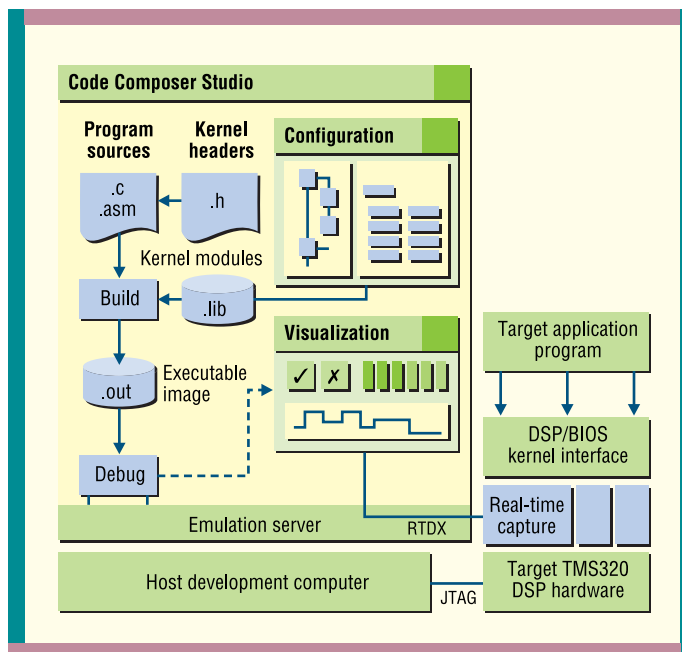


Figure 1. Real-time kernels have entered the DSP scene as a result of the growing number of tasks assigned to the target. For instance, application programs may engage simultaneously in digital signal processing and general-purpose system control functions; handle incoming events from multiple external sources; manage all kinds of ports, timers, DMAs, and host interfaces; move data; produce more than one data stream; execute algorithms at different frame rates—and even more. TI’s DSP/BIOS kernel essentially is a library of functions callable from C or assembly language. A palette of kernel modules can be included in a mix-and-match fashion within an application program.

sor or microcontroller. By eliminating the controller, designers save money and power; need only one set of tools instead of two; and can develop one code base, not two.

A standard kernel and advances in compiler technology as well as the natural ability to express functionality in a high-level language make such multitasking possible (Figure 1). Moreover, when such a kernel is highly integrated with the rest of the tooling, developers inherit some uncommon debugging and diagnostic capabilities—a welcome event considering that integration and debugging time have, in the words of one industry observer, “exploded to become by far the largest portion of any development.”

Take Blue Wave Systems, Inc. of Carrollton, Texas. A leading supplier of high-channel DSP subsystems used in telecom infrastructure equipment, such as VoIP gateways, digital wireless communications, and intelligent peripherals, it’s made Texas Instruments’ scalable, extensible kernel, DSP/BIOS II, the core infrastructure of its recently unveiled fax framework. Called ComStruct, the framework is actually a telephony development environment. The ComStruct voice- and fax-over-IP solution provides more than 120 channels of voice and fax relay or 120 channels of modem in an integrated deployment platform.

The escalating power of DSP hardware has led not only to the running of multiple algorithms, but also to a search for better compiler technology. One reason: the need to slash code size as applications continue to balloon in size and complexity.

“The embedded arena is one of the few places left where people care about the size of an executable program, because they pay for it to go into ROM,” says

The escalating power of DSP hardware has led to the running of multiple algorithms.

Keith Cooper, a professor and researcher at the Department of Computer Science at Rice University in Houston, Texas.

Compiler technology offers opportunities to reduce code size. For example, according to Cooper, many traditional compiler optimizations are designed to reduce the execution time of compiled code, but not necessarily the size of the compiled code.

Because much of the code for embedded systems is compiled once and then burned into ROM, the soft-

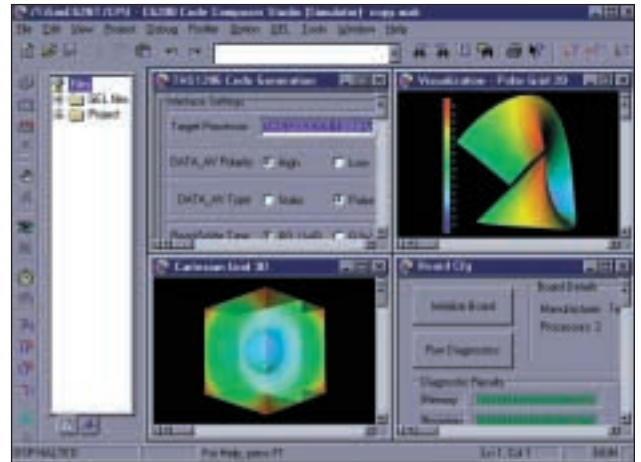


Figure 2. Users are asking for integrated development environments that provide an open, “pluggable” architecture. That is, third-party tool developers can produce plug-ins—software applications that use the environment’s application interface and are integrated with, customize, or extend the host and target development environment with additional, specialized functionality. Shown here are some typical plug-ins that can be integrated with and launched from Code Composer Studio: a TI data converter configuration and set-up tool (top left), a 3-D data visualization and analysis tool (top right and bottom left), and a configuration and diagnostics tool for a development board (bottom right).

ware designer will often tolerate much longer compilation times as a trade-off for smaller compiled code size.

Cooper’s approach is to take advantage of that by using a genetic algorithm, essentially a biased sampling search technique, to find optimization sequences that generate small blocks of object code, then compare the solutions to those found by using a fixed optimization sequence and by testing random optimization sequences.

Based on the results found by the genetic algorithm, the researchers develop a new fixed sequence to reduce code size. They also explore the idea of using different optimization sequences for different modules and functions of the same program.

SHORTENING THE WAIT

As that kind of work continues, commercial compilers have come down in some corners of the embedded world—that is, they are shaving lines of code from exe-

Available Now!

Introducing Our New USB-JTAG Emulator



- **USB (bus powered) Peripheral • Portable and Compact (1.1 x 2.6 x 5.5 inches)**
- **Works with TI Code Composer Studio**
- **TMS320C5000 and C60000 Support**
- **Quick Installation**

DSP Enabling Technologies™

- **Development Hardware**
- **Operating Systems**
- **Bundled Toolsets**
- **Design Services**
- **Consulting**

Blackhawk
by EWA Inc.

© 2001 EWA

See us at **booth #131** at the Embedded Systems Conference - San Francisco 2001

For more information please visit: www.blackhawk-dsp.com or phone: 1-877-983-4514

cutables. But perhaps just as important, they're cutting the lag time between new DSP architectures and compiler technology. Instead of the five-year wait typical in the mainstream commodity processor arena, C compilers for DSP development finally are beginning to get a jump on new architectural features, a boon for those who want to get to market fast with the latest hardware.

More accurately, the more enlightened semiconductor companies design advanced DSP functions with C coding in mind or design compilers and chips together. A case in point: the high-performance TI C6000 platform.

"We decided years ago to prepare for both a large leap in DSP performance and C efficiency," says Rich Scales, a manager in TI's Software Development Solutions Operation in Houston and an expert on compilers. That early insight may explain the reaction of Delphi's Kane when he says, "There's a lot more formalism and less seat-of-the-pants feel to DSP software over the last two years."

In fact, leading companies such as Cisco, Ericsson, Nokia, and 3Com—all of which want to reuse code in rather large diverse applications—are insisting on efficient C compilers. A profile-based compiler goes a long way toward satisfying that need.

Basically, such a compiler provides users with a

C compilers for DSP development are getting a jump on new architectural features.

graphical way of trading off code size for performance in their applications. The technique may be best applied in parallel processing, which can rocket in code size because of performance-boosting techniques such as aggressive unrolling and software pipelining.

In fact, in many applications, 80 percent of the con-

trol code falls into 20 percent of the nets. Instead of users' trying to optimize all control code, the profile-based compiler analyzes all of the code and then profiles, in a two-dimensional graph, the various options falling between the high-performance and small-code-size ranges. Somewhere along that graph is a knee representing a significant code size decrease with only a modest performance hit.

Some compilers go much further. Possessing the knowledge of the expert hand-coded assembly writer, they provide interactive tuning and feedback, using an understanding of an entire application to optimize key components.

Just around the corner are a host of other compiler assets that will allow users to measure, tune, and build applications based on a myriad of important parameters, such as cycle

counts, cache hits and misses, and memory locations.

SOFTWARE DEFINES THE RELATIONSHIP

Compilers, kernels, algorithm standards: How do these relate to one another? Each has become a crucial part of a contemporary integrated development environment—a central nervous system, if you will, not only for differentiating products. In other words, not all Internet audio devices, for example, are alike; software differentiates one from the other, and the development environment that launched the winner takes much of the credit.

Such an environment, that can produce winning "X generation" products and reuse diverse software components across multiple sites, multiple developers, and even multiple companies, goes beyond the usual lineup of individual tools: It calls for heavy integration into a cohesive continuum. It also calls for the ability to accept plug-ins, compliant tools from diverse sources that use the IDE's application interface and are integrated with, customize, or extend the host and target development environment with additional specialized functionality (Figure 2).

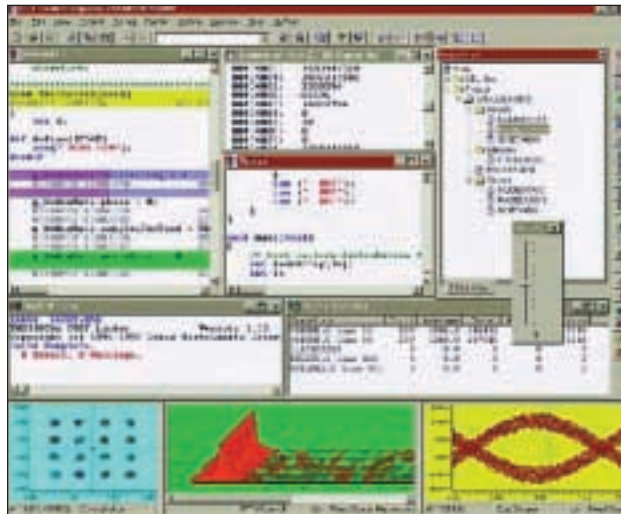


Figure 3. Besides plug-ins, the latest development environments help boost productivity with advanced features, such as real-time analysis and the ability to visualize activity without stopping a processor, as shown here.

Compilers, kernels, BIOS, algorithm standards, debuggers, emulators, linkers, editors—all, then, must dovetail into a cooperative framework that users can put to work in optimizing their code. For example, in TI's vision a compiler would be aware of, and ready to support, TMS320 DSP Algorithm Standard components or stand ready to let users juggle code density and MIPS.

Like the compiler, other traditional components of the framework are undergoing plastic surgery. For example, facelifts for linkers and other tools are moving decidedly toward the visual.

The move away from cumbersome text-based languages should be appreciated by those who use extensive memory system overlays, different run-time and load-time addressing or code location trials, and the like. With a visual linker, all of those can be accomplished graphically merely by clicking and dragging various memory components—programs or data—and putting them into a graphical view of the memory

space. Not only can users drag and drop components into multiple memory types, but they also obtain immediate visual feedback and can choose from a library of standard devices.

What of the future? Can DSP development software get any better? Of course it can. Users are asking for many things, including even tighter integration with the chip of choice, larger file sizes, better ways of visualizing data (Figure 3), improved editors and project managers, integration of favorite tools—and the list goes on.

Look for gains in all of those areas, and more. Think, for instance, about real-time analyses, especially enabled by the BIOS; configurable editors; expanding libraries to support chips, boards and peripherals.

Users of TI's eXpressDSP technology or Code Composer Studio also can expect a deluge of third-party plug-ins that will make their development life a lot easier in a variety of application areas, from specific communications functions to general-purpose data converters. ◆

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Mailing Address:

Texas Instruments
Post Office Box 655303
Dallas, Texas 75265