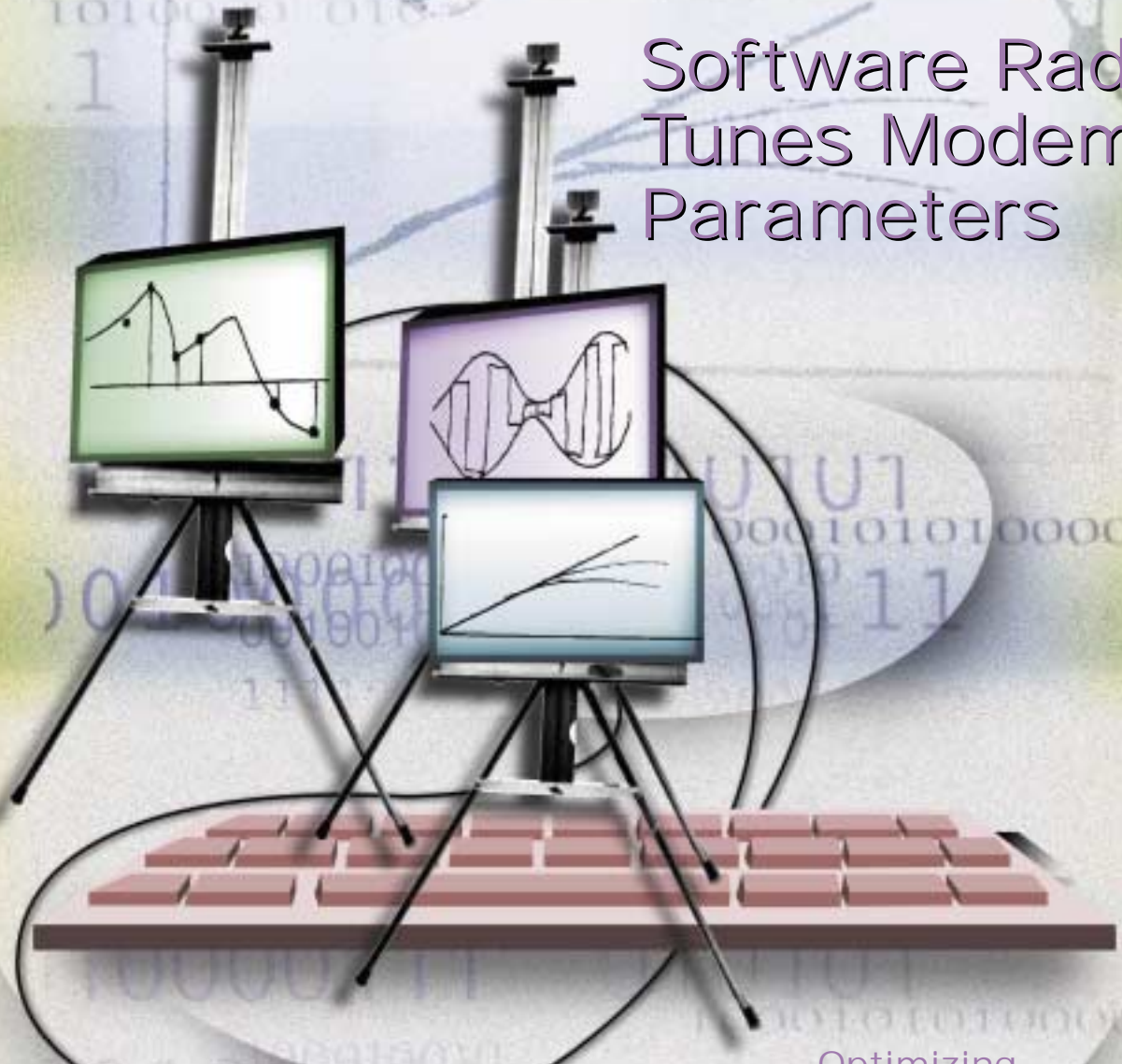


Practical solutions for DSP system developers

Embedded Edge

June 2001

Software Radio
Tunes Modem
Parameters




Optimizing
Soft Modems

New Tools Speed
the Development
of Multi-DSP
Applications

The right emulator won't leave you stranded when new DSPs come along.



flexds

With new TMS320 DSPs coming out with dizzying regularity, you need a development platform that stays up-to-date. Only the ultra-flexible FlexXDS from DSP Research lets you keep up with Texas Instruments! FlexXDS is a new breed of emulator – a reconfigurable platform that expands to become a complete hardware and software development system and a powerful test bed for your TMS320 applications. No one has more experience with TI DSP development than DSP Research. So when TI puts you to the test with a new DSP, look to FlexXDS for answers!  **DSPR**

“DSP Research, with its FlexXDS product family, continues their strong commitment by keeping up with TI’s torrid pace of introducing new products.”

♦ Mark Mattson, Texas Instruments marketing manager

www.flexds.com/offer
phone: 1.408.773.1042



Key FlexXDS Features:

- PCI-based in-circuit emulator
- JTAG POD with detachable cable and diagnostic LEDs
- DSP and I/O Expansion modules for TI's C6000, C5000 and more
- More than 15 modules available
- Debug software before your target system is ready



Embedded Edge

A Texas Instruments Publication

Volume 2 June 2001 Number 2

Stan Runyon
Editor-in-Chief
 testman2@earthlink.net

Mike Robinson
Managing Editor
 mrobinso@cmp.com

Tim Moran
Creative Director

Donna Moran
Art Director

Genevieve Joerger
Director, Custom Solutions
 gjoerger@cmp.com

Gregory Montgomery
Director of Sales
 gmontgom@cmp.com

Grace Adamo
Project Manager

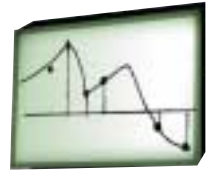
Robert Steigleider
Ad Coordinator
 rsteigle@cmp.com

Susan Harper
Circulation Director
 sharper@cmp.com

Embedded Edge is published by Texas Instruments, Inc. and produced in cooperation with CMP Media Inc. Entire contents Copyright © 2001. The publication of information regarding any other company's products or services does not constitute Texas Instruments' approval, warranty or endorsement thereof. To subscribe on-line, visit: www.edn.com/customsolutions/edge/subscribe.ftml

Code Composer Studio, TMS320, TMS320C6000, C6000, TMS320C5000, C5000, TMS320C2000, C2000, DSP/BIOS and eXpressDSP are trademarks of Texas Instruments, Inc. All other trademarks are the property of their respective owners.

Inside This Issue



Insighter: The Power of Cheese 4
The power of DSP capabilities can produce startling wonders.

Breakpoints 6
News from the providers of embedded systems development products and services.

Cover: Software Radio for Wireless Internet 8
With the latest DSP devices, you can design a software-configurable, low-cost modem that's very efficient and small.

Optimizing C-Baseline Modems 16
In just a few short phases, you can optimize C-coded reference modems to meet higher-density targets.

Speeding Development of Multi-DSP Apps 22
New software tools are taking aim at multiprocessor DSP systems, particularly for the TMS320C6000 DSP platform.

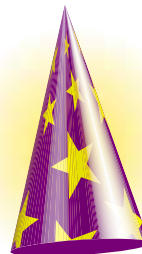
Profiler Boosts Code Optimization 30
A software-based statistical real-time profiler can help you dramatically improve the performance of TMS320C62x DSP code.

Wizards' Corner 34
Answers to developers' questions from experts in embedded systems development.

Launchings 35
New products and services for embedded systems developers.

On the Edge 38
Needed: New compilation tools to help optimize embedded code.

33



34



38

The Power of Cheese

We've all seen the commercial: A precocious waif leaves a plate of cheese for Santa instead of milk and cookies, and lo and behold, Santa leaves behind a roomful of luxury goods, ranging from top-line cars to high-tech electronics. The commercial concludes with the punch line rolling across the screen: "Behold the power of cheese."

Clever, the American Dairy Associations. Too bad the semiconductor industry doesn't have the equivalent to similarly promote DSPs, for the power of DSP capabilities can produce equally startling wonders.

Take modems for wireless Internet devices—a technical challenge if there ever was one. But base wireless functions on digital signal processing and all of the advantages of the digital domain accrue, and new algorithms eager to exploit those advantages are appearing.

Basically, the design challenge is to achieve robust communications at high data rates—10 to 100 Mb/s wouldn't be unusual—and decent spectral efficiency. Naturally, minimal setup time, low maintenance, and a competitive price go along, and provision for emerging frequency bands.

Those goals are achievable, says Andrew Bateman, the author of our cover story. Indeed, the power of modern DSP devices and software can comfortably embrace source and channel coding, pulse shaping, modulation, demodulation, quadrature frequency translation, power amplifier linearization, receiver dynamic range extension, calibration of the in-phase and quadrature signal components, automatic power and frequency control, and direct digital synthesis. Unfortunately, there isn't enough room inside to focus on everything, but you'll get a good idea of how DSPs and software can be tapped for pulse shaping, demodulation, and PA linearization.

Are you more interested in using C-coded reference modems? Want to stuff as many as possible onto a single chip without assembly coding? Ah, the power of Code Composer Studio's optimizing compiler for DSP platforms, especially when it's tickled to up the density. How dense? According to Commetrex, which went through the optimizing process, C-baseline modems can soar from 6 per 200-MHz DSP chip to 28 in four short project phases. Need even more? A fifth phase

can take the number of channels to 48 per DSP. Get the details from Ghassan Farah, who shows you exactly how to go about it yourself.

As powerful as DSP chips are, bring them together and stand back—your application is liable to take off on you. OK, not quite, but new software tools—frameworks—for multiprocessor DSP systems promise much faster development of application software, plus a slew of other benefits.

Each one has its own claim to fame, but generally the tools are aimed at sorting out some common challenges with multiprocessors—partitioning of the problem and its data, processor communications and synchronization, simulation, and debugging. Fiona Culloch, from 3L, describes the two types of frameworks—ones based on writing C code and graphical development environments, with examples, and details how the former works.

While you're beefing up your DSP muscle, you could deploy a software-based statistical real-time profiler to dramatically boost the performance of your application even more. According to Konstantin Merkher and Jacob Bridger, of Surf Communication Solutions, optimizing code to boost performance is one of the greatest challenges in writing real-time DSP code. But if you write a real-time profiler, you can do it—to the tune of several orders of magnitude. Learn how inside.

More power to DSPs!

—Stan Runyon
testman2@earthlink.net



DSP DESIGN TOOLS

THE LEADER IN PORTABLE EMULATORS FOR TI DSP'S

We have your high performance tools for your DSP projects

- **Emulators:**
XDS510PP PLUS, SPIS10, SPIS15, SPIS20, SPIS25, SPIS30, SPIS40
- **Debuggers:**
TI Code Composer/Studio, Allant ASPEX
- **Evaluation Modules:**
CSX, C203/F206, F240/F243/LF2407, C548/VCS49/VCS402/VCS409/VCS410/VCS416/VCS472
- **eZdsp™ and DSks:**
F240/F243/LF2407, eZdigs for LF2407/VCS402/VCS410/VCS33
- **Software Tools:**
'C' compilers, assemblers, linkers
- **Operating Systems:**
RTXC, Real Time Executive in 'C'
- **Miscellaneous:**
Digital Motor Controllers, Surround Sound Modules, Prototype Modules, Power Supplies

Dr. DSP
Says



**Spectrum Digital =
Tools for
DSP Development**
QED

**SPECTRUM
DIGITAL**

INCORPORATED

12502 Exchange Drive, Suite 440
Stafford, Texas 77477

T: 281.494.4505

F: 281.494.5310

www.spectrumdigital.com

• AUSTRALIA: 61 3 9720 5344 • BELGIUM: 31 20 351 20 91 • BRAZIL: 55 11 452 5380
• FRANCE: 33 547 672954 • GERMANY: 49 2033 570977 • INDIA: 91 80 6636333 • INDIA:
91 80 2245837 • ISRAEL: 972 3 7657640 • JAPAN: 81 3 5823 0191 • KOREA: 82 2 783 8655
• KOREA: 82 2 409 7277 • PE OF CHINA: 86 810 6847 1495 • POLAND: 48 22 724 30 39
• SCANDINAVIA: 46 40 45 95 70 • SINGAPORE: 65 844 9789 • SWITZERLAND: 41 3197 22152
• TAIWAN: 866 2 2881791 • UNITED KINGDOM: 44 1484 351006

TI Third Party Network
Member



eXpressDSP Compliance for Sound, Speech Techs

WOW audio enhancement technology from SRS Labs, Inc. (Santa Ana, Calif.; www.srslabs.com) has passed eXpressDSP compliance testing on the TMS320C54x and TMS320C55x DSPs. The technology, which is embedded in Microsoft's Windows Media Player 7, improves the dynamics and bass performance of stereo audio played through small speakers and headphones. The



announcement follows a similar one regarding SRS Labs' Voice Intelligibility Processor (VIP) technology, which raises the quality and intelligibility of speech in voice communications equipment and speech synthesis equipment, such as conventional and cellular phones, VoIP devices, headsets, and digital radios. VIP's small footprint fits easily on top of existing voice coder or processor applications.

Linux kit for TI DSPs debuts

RidgeRun, Inc. (Boise, Idaho; www.ridgerun.com) has unveiled the DSPLinux Software Development Kit (SDK) for the OMAP1510 and TMS320DSC21 processors from Texas Instruments, which combine an ARM7 or ARM9 core with the TMS320C5000 DSP platform. Now available in beta, the kit includes a 2.4 version of the Linux operating system optimized for embedded devices, standard GNU development tools, and the Desktop Simulation Environment.



Mentor, TI Team for Coverification

Mentor Graphics Corp. (Wilsonville, Ore.; www.mentor.com) and Texas Instruments, Inc. (Dallas, Texas; www.ti.com) have agreed to deliver coverification Processor Support Packages (PSPs) for TI's DSPs and microcontrollers. The PSPs are based on TI's present instruction set simulators and connect to Mentor Graphics' Seamless Co-Verification Environment through an adapter layer. PSPs included in the agreement model the TMS320C27x, C54x, C55x, and C2000 DSPs, as well as the ARM925 microcontroller core. They work with all popular logic simulation platforms and are compatible with Mentor's library of PSPs offered by for use in multiprocessor systems.



Blue Wave Systems to Join Motorola's Computer Group

Blue Wave Systems (Carrollton, Texas; www.bluews.com), a long-standing TI DSP Third Party Network member, and Motorola, Inc. (Schaumburg, Ill.; www.motorola.com) have signed a definitive merger agreement in which Blue Wave will join the telecommunications business of Motorola's Computer Group (Tempe, Ariz.; www.motorola.com/computer).

Best known for its ComStruct software environment, which includes the use of DSP/BIOS as well as integrated eXpressDSP-compliant algorithms, Blue Wave will continue its operations in Carrollton and Loughborough, U.K.

I-Logix Plots Software Component Strategy

I-Logix, Inc. (Andover, Mass., www.ilogix.com) has rolled out a three-phase plan to build a comprehensive development platform for writing and reusing seamless, component-based embedded software. The strategy, which will unfold during the course of the year, will let developers snap existing software components into their designs, similar to the way that IP building blocks are assembled to construct systems on chip and other ICs.



The first phase, available now in Rhapsody 3.0, lets developers "componentize" and reuse legacy software modules that can be viewed within the UML graphical model. The second phase will equip Rhapsody with an intuitive visual metaphor for assembling model-based executable code components into embedded real-time applications.

In the third phase, the company will provide a Web-based structure to organize and catalogue software components, encouraging the exchange of design information. As part of the plan, I-Logix will also integrate iNOTION product life-cycle technology into its existing products. The technology, acquired in March from KLA-Tencor (San Jose, Calif.; www.kla-tencor.com), will let embedded development teams store, support, and maintain design components in a central repository.

More Breakpoints on page 33

FREE
PRODUCT
TRIALS
ONLINE

MATLAB and Code Composer Studio.

Together at last.

The leaders in DSP development now work together. With the new



The Developer's Kit for TI DSP works with C5000[™] and C6000[™] platforms.

Developer's Kit for Texas Instruments DSP from The MathWorks, you can execute Code Composer Studio[™] functions in MATLAB to interactively debug applications, analyze real-time data, and automate testing on TI DSPs. And you can streamline implementation by generating

real-time prototypes and Code Composer Studio projects from Simulink.

Demos, tutorials, and trials are available online.

See the fastest way to develop, test, verify, and implement DSP software. Visit www.mathworks.com/eme.

MATLAB[®]
& SIMULINK

Debugging
Data transfer
Real-time analysis
Algorithm verification
Test automation
Code generation



The MathWorks DSP tools accelerate the development of wireless, broadband, telephony, and multimedia products.

The MathWorks

Visit www.mathworks.com/eme
or call 508-647-7040

Visit our web site for information on training courses online, on site and at public locations around the world.

All trademarks are property of their respective owners.



© 2011 The MathWorks, Inc.

CONNECTING WIRELESS INTERNET DEVICES USING DSP SOFTWARE

With the latest DSP devices, you can design a software-configurable, low-cost modem that's very efficient and small.

By Andrew Bateman

It's a very exciting time if you're designing a wireless modem. The Internet is the fastest-growing sector of the IT market, and high-speed wireless connection for business and residential users is a huge business opportunity. You also have unprecedented flexibility in the range of wireless functions that can be implemented in the digital domain, and new algorithms that fully exploit this advantage are appearing.

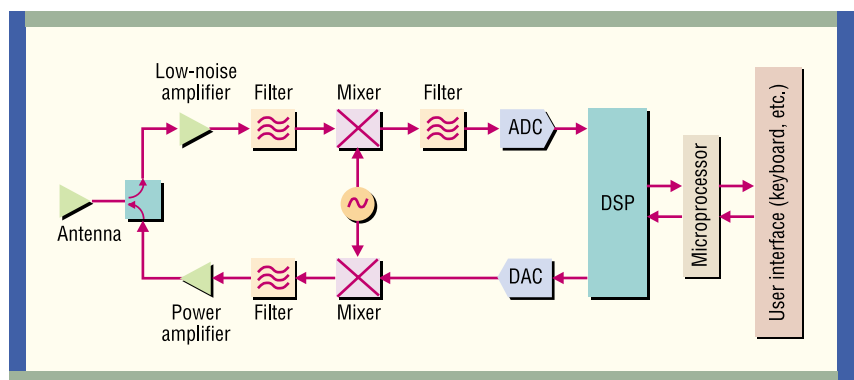
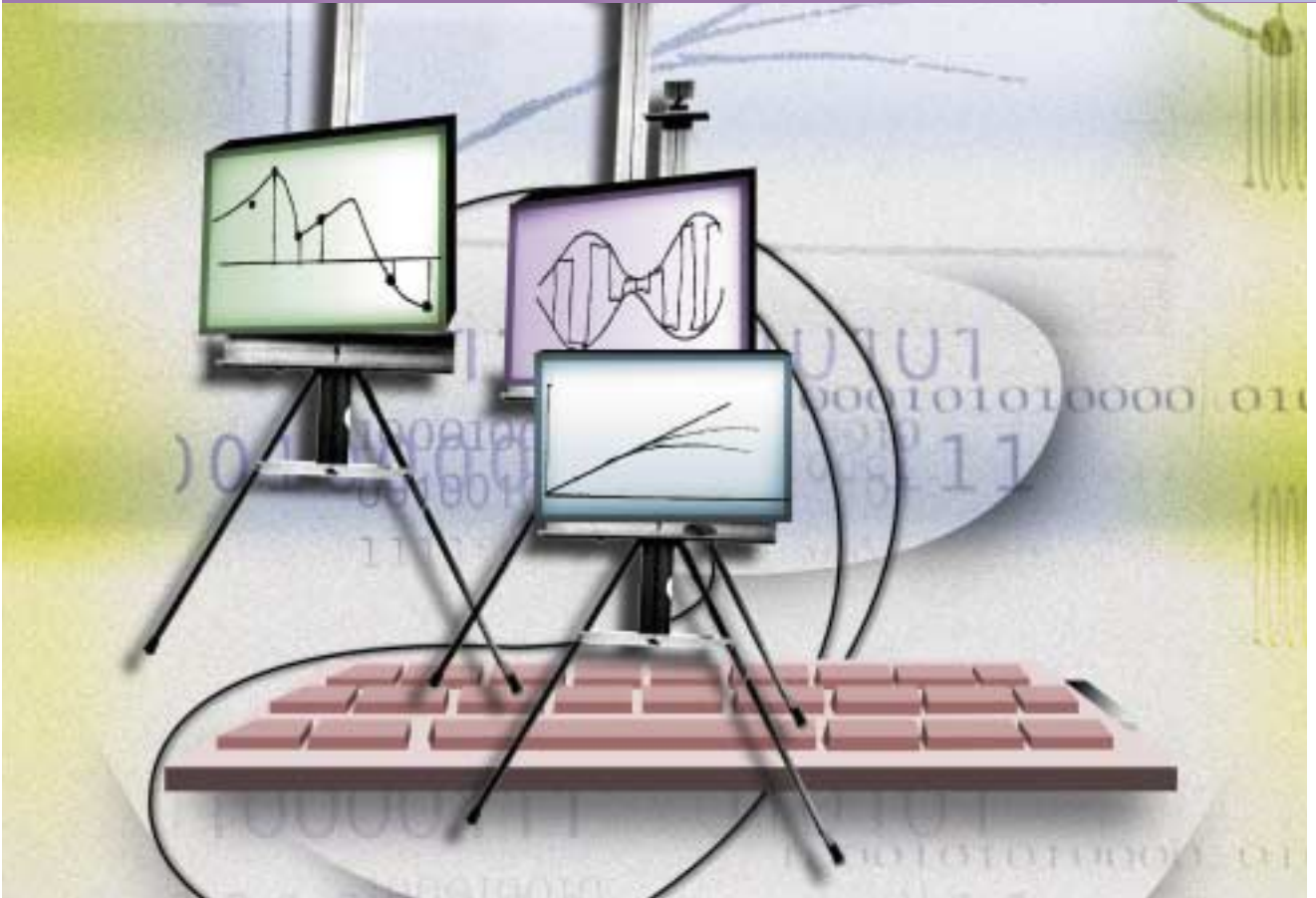


Figure 1. The architecture of a modern software radio is surprisingly simple, comprising only a few core building blocks. Linearity in the power amplifier and a good low-noise front end and synthesizer are essential, as of course are a high-speed, low-cost, low-power DSP engine and data converters (highlighted).



Still, wireless modems present a serious technical challenge, and that's even truer for wireless LAN (WLAN) devices. Cost-effectively distributing the ultrahigh capacity of a fiber node (multiple gigabits per second) to disparate users often requires a radio or free-space optical transmission link. Data rates in excess of 10 Mb/s are called for, with upward of 100 Mb/s a target for larger business users. In addition, a point-to-multipoint network or a distributed network is needed for distribution from a fiber hub.

For the WLAN terminal, your design challenge is to achieve robust communications at those high data rates, with good spectral efficiency (to maximize the number of customers that can be served for a given frequency allocation). Of course, minimal setup time, low maintenance, and a competitive price are assumed. From a manufacturing standpoint, the wireless platform must be easily tailored to new frequency bands as they become available and be able to flexibly exploit and manage the characteristics of the channel and variable data transfer demands.

Those goals are now achievable. By harnessing the ever increasing performance/power potential of modern DSP devices such as the Texas Instruments TMS320 DSP family, you can build a highly configurable, low-cost solution with minimal frequency-

selective components, high efficiency, and small size.

Unlike the cellular phone market, which is tightly controlled by standards—GSM/IS95 and third-generation (3G) Universal Mobile Telephony Service (UMTS)—the WLAN market has no dominant air interface standard. Literally dozens of proprietary systems are in play. This situation gives you the flexibility to work with the latest modulation, coding, access, and equalization formats, thus squeezing every last drop of capacity out of the channel. However, the manufacturer has to build a degree of flexibility into its design (or develop multiple versions) so that the product can operate with multiple air interface standards across a range of service providers and network types.

Achieving flexibility in a wireless platform requires three key features: minimal frequency-selective components, high linearity to minimize self-induced signal distortion, and maximum software-defined functionality.

The core building blocks of a modern software-defined digital radio are a linear power amplifier (PA); a wideband, low-noise front end; a good synthesizer; and, of course, a high-speed DSP engine and data converters (Figure 1). For maximum software configurability, the analog signals should ideally be converted into digital form at the RF carrier frequency. Unfortunately,

the conversion isn't viable at frequencies much above 500 MHz; with most WLAN spectrum allocations in the 5-GHz-plus range, an analog intermediate-frequency stage is still required. The digitization frequency used is governed by component costs; the linearity, speed, and dynamic range of D/A and A/D converter technology; and power consumption constraints.

Both the linearity and the speed of converters have improved significantly, with 14-bit, 100-megasample-per-second, 80-dB SNR converters readily available. These devices can easily support 40-MHz subsystems using direct sampling and can extend toward 300-MHz IF solutions using subsampling methods.

The power consumption of fast converters is significant (several hundred milliwatts), and further reduction of IC feature size or other advances in process technology are required to realize the power savings needed to allow IF sampling in handheld equipment. For fixed WLAN installations, however, power consumption is less critical; instead, linearity and sampling rate are likely to dominate the choice of frequency.

DSP tasks for a high-speed wireless Internet modem comprise core modem functionality (source and channel coding, pulse shaping, modulation, demodulation) and more advanced software radio management tasks (quadrature frequency translation, PA linearization, receiver dynamic range extension, calibration of the in-phase and quadrature [I-Q] components of the signal, automatic power and frequency control, and direct digital synthesis, for example). We'll focus here on pulse shaping, demodulation, and PA linearization.

PULSE SHAPING

To maximize the data transmission rate over a wireless link with finite bandwidth, you must shape the data pulses modulating the carrier signal. For modems using frequency shift keying (FSK), shaping traditionally involved Gaussian filtering; for the more advanced quadrature amplitude modulation (QAM) modems, a root raised cosine (RRC) filter is commonly employed. Whatever the pulse shaping, there are two core algorithms for implementation: the classical filter and the lookup table (LUT) method.

The filter approach is normally realized using an infinite impulse response (IIR) filter for a Gaussian filter shape and a finite impulse response (FIR) filter for the RRC shape. The FIR filter realization allows the synthesis of a near-perfect RRC transfer function with linear phase and controlled stopband attenuation. (IIR approximation of an RRC filter can be used, but in the

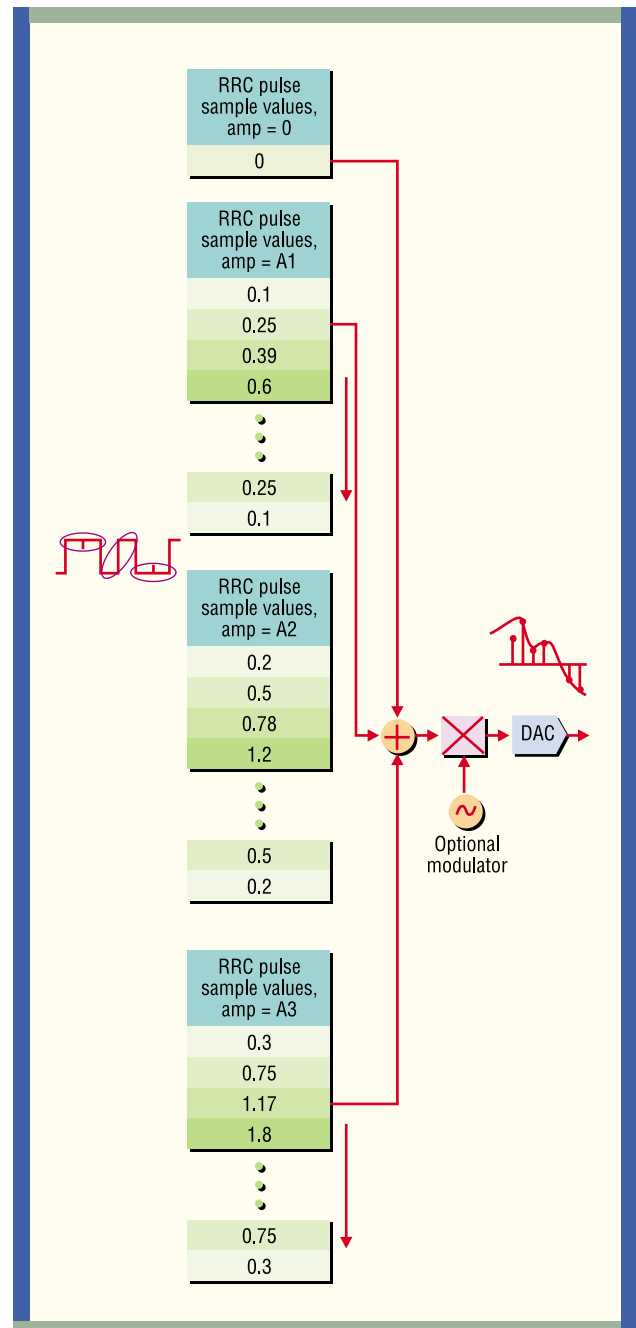


Figure 2. Although appearing more complex at first glance, the lookup table (LUT) method of symbol pulse shaping, using the data transitions to index the stored RRC pulse shape, imposes little processing overhead. The output waveform is synthesized from the summation of the pulse samples for several preceding data bits.

majority of cases, the FIR structure yields a lower-overhead algorithm than the compensated IIR design.) The RRC filter shape is always an approximation of the true Nyquist filter response, trading off filter length (and hence delay and processor load) for stopband attenuation and roll-off rate. Most filter design packages offer raised cosine (RC) and RRC filter options, making it easy to explore the trade-off between the two parameters.

In many cases, it may be preferable to cascade an RRC filter with a second, FIR filter (possibly half-band for ease of implementation). The second filter achieves the desired level of out-of-band attenuation to meet a given spectral mask while relaxing the requirements on the first—and more processor-intensive—filter.

The alternative approach to pulse shaping makes use of a lookup table. The table holds the precalculated values for the pulse response of the desired filter, based on all possible input state transitions (Figure 2). The state transition is used to index the correct stored pulse response from the chosen filter, which is then


summed with the pulse responses from previous transitions to form the composite pulse-shaped waveform.

The lookup table method is preferred when execution time is at a premium, as lookup table indexing carries very low overhead. Conversely, the real-time filter realization of pulse shaping is used when memory space is at a premium and storage of the multiple filter pulse responses is impractical.

DATA DEMODULATION

Efficient algorithms for data demodulation are key to the success of software-defined wireless modems. A broad range of demodulation algorithms are in widespread use, mirroring the multiple modulation formats, coding strategies, carrier and symbol timing recovery mechanisms, and equalization methods deployed.

We'll look at one demodulation algorithm for frequency discrimination that has widespread application and is a good example of a DSP-optimized solution. Frequency discrimination has two primary uses:




Looking for Mr. Right?

Single Board Computer, strong, independent, flexible, green, 160mm x 100mm, USB, 150 MHz DSP, 2 OMNIBUS sites seeks long term relationship with intelligent end-user. Call anytime 805-520-3300.

Download the SBC6711 with multiple OMNIBUS modules to customize for your embedded solution!

	Analogue Input	Analogue Output
A4D4	4 Ch. 16-bit @ 200 kS/s	4 Ch. 16-bit @ 200 kS/s
A4D1	4 Ch. 12-bit @ 10 MS/s	1 Ch. 16-bit @ 10 MS/s
A16D2	16 Ch. 16-bit @ 100 kS/s	
AD16	8 Ch. 12-bit @ 40 MS/s	
AD40	4 Ch. 10-bit @ 7.5 MS/s	
AD1		4 Ch. 16-bit @ 200 kS/s
RF	8 Ch. 12-bit @ 40 MS/s	7 Ch. 12-bit @ 40 MS/s
SD	4 Ch. 24-bit @ 2 MS/s	4 Ch. 24-bit @ 2 MS/s
Servo16	16 Ch. 16-bit @ 4 MS/s	16 Ch. 16-bit @ 4 MS/s

OMNIBUS Modules



www.innovative-intg.com
905.520.3300 phone • 905.579.3740 fax


Featu

- ▶ 150 MHz 160220/01/11 floating-point DSP
- ▶ Operates independently from a host PC
- ▶ Plug-n-Play USB interface
- ▶ Dual OMNIBUS I/O Module sites
- ▶ Comprehensive C/C++ cross development tools & servo algorithm templates
- ▶ Windows 9x/NT/2000 drivers

Applications

- ▶ Noise Cancellation
- ▶ Embedded Control
- ▶ Wide-Channel Audio
- ▶ Precision Motion Control

SBC6711



demodulation of the FSK family of waveforms, such as Gaussian minimum shift keying (GMSK) used in GSM cellular communications, and automatic frequency control loops. The DSP frequency discriminator algorithm uses differentiation and cross multiplication to generate an output that directly corresponds to the instantaneous frequency of the input signal samples. An optional envelope normalization block removes fading components present in the input signal (Figure 3).

Mathematically, the operation of the quadrature discriminator is as follows: For general complex input signals of the form:

$$I(t) = \frac{1}{2} r(t) \times \sin \phi(t)$$

$$Q(t) = \frac{1}{2} r(t) \times \cos \phi(t)$$

where $r(t)$ is the signal envelope and $\phi(t)$ is the angular phase/frequency, the signals at the outputs of the two differentiators can be represented as:

$$\dot{I}(t) = \frac{1}{2} r(t) \times \dot{\phi}(t) \times \cos \phi(t) + \frac{1}{2} \dot{r}(t) \times \sin \phi(t)$$

$$\dot{Q}(t) = -\frac{1}{2} r(t) \times \dot{\phi}(t) \times \sin \phi(t) + \frac{1}{2} \dot{r}(t) \times \cos \phi(t)$$

By cross-multiplying and subtracting the signals as shown in Figure 3, you obtain an output signal, given by:

$$\dot{I}(t) \times Q(t) - I(t) \times \dot{Q}(t) = \frac{1}{4} r^2(t) \times \dot{\phi}(t)$$

Further division by the (envelope)² term yields a normalized real-time measure of the instantaneous frequency variations of the input signal. (In practice, it's much more efficient to use a lookup table to generate $1/r^2(t)$, which is multiplied with the top path signal.)

Unlike conventional frequency discriminators based on a phase-locked loop (PLL), the algorithm doesn't involve a feedback process. It also introduces little or no bandwidth expansion into the signal, thus ensuring that the Nyquist sample rate limit is not violated.

Don't be left hanging...



...without the right DSP tools for your project!

Ensure your success with
Ultimate Solutions
 Your reliable source for professional
 DSP development tools!

Ultimate Solutions, Inc.
Development Tools for Embedded Systems

2 Oak Drive, Hampton Falls, NH 03844
 Telephone: 603-929-9855 • Fax: 603-696-6353
 Email: info@ultsol.com • Web: http://www.ultsol.com





Perfect for your environment... Blackhawk™ USB-JTAG Emulator Now Supporting C2000 DSP processors

- USB (bus powered) Peripheral • Portable and Compact (1.1 x 2.6 x 5.5 inches)
- Works with TI Code Composer Studio™
- TMS320C5000 and C6000 Support
- Quick Installation

DSP Enabling Technologies™

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

Blackhawk ™

by EWA Inc.

© 2001 EWA

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

Code Composer Studio™ is a registered trademark of Texas Instruments

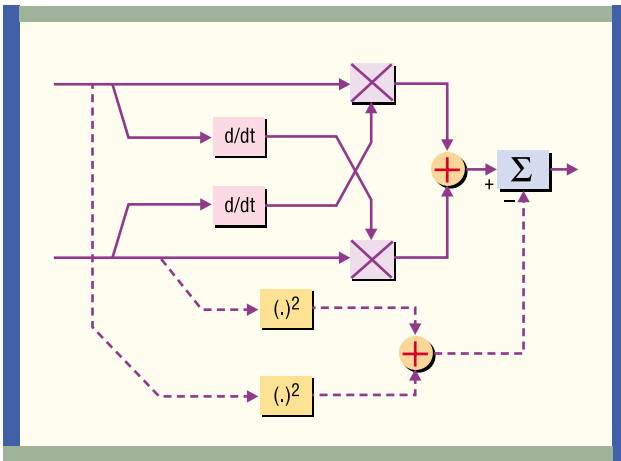


Figure 3. Obtaining an instantaneous measure of the frequency of a signal is very difficult using analog methods. In contrast, the algorithm for frequency discrimination shown here uses quadrature processing to recover the instantaneous (normalized) frequency of the input signal with very low processing overhead.

PA LINEARIZATION

Pulse shaping and multisymbol modulation are wasted if the waveform is overly distorted after passing through the analog transmitting or receiving functions. Designers often sacrifice linearity because they're trying to maximize PA efficiency and power output, so some distortion often results. The conventional solution—backing off the PA drive to operate within a linear portion of the PA characteristic—wastes considerable power. It's much more efficient to use DSP techniques to predistort the waveform of the signal driving the PA, in a complementary manner to the PA nonlinearity.

The source waveform is passed through a lookup table, which stores the correction factor for the amplitude (and phase) of the waveform at any given PA drive level (Figure 4). For most applications, the PA characteristic isn't stable enough (for example, temperature, supply voltage, output load) for the process to be purely open-loop. A feedback path from the PA output is therefore usually employed to allow the residual PA distortion to be measured and the lookup table coefficients to be updated.

For the high-bit-rate solutions required in WLANs and the LUT-specific nature of the processing tasks involved, it's often best to implement adaptive predistortion using dedicated ASICs or FPGAs.

A/D AND D/A CONVERSION

Many manufacturers, including Texas Instruments, are competing aggressively in the area of high-speed digital IF converter solutions, and new devices are appearing on the market almost weekly.

Digital up and down conversion. The initial task for the digital processing unit is to convert the IF signal into a complex baseband form (down conversion) and from complex baseband to IF (up conversion). Down conversion ensures minimum sampling rate processing for the remaining radio functions. The tasks involve mixing (multiplication) with quadrature versions of a digital oscillator. In addition, a process of interpolation and decimation is needed in the up converter and down converter, respectively, to optimize the sampling rate between the digital IF requirements and the complex baseband requirements.

Because the two functions (frequency mixing and sample rate conversion) are common to all digital IF solutions, various manufacturers have produced dedicated ICs optimized for digital up and down conversion. The high-sample-rate processing associated with the digital-to-analog interface can be accommodated in these hardwired devices, allowing the (comparatively) slower digital signal processing of the wireless signal content to be undertaken in cheaper, lower-power, software-programmable DSPs.

An alternative route that maintains full flexibility of design is to implement the mixing and sample rate

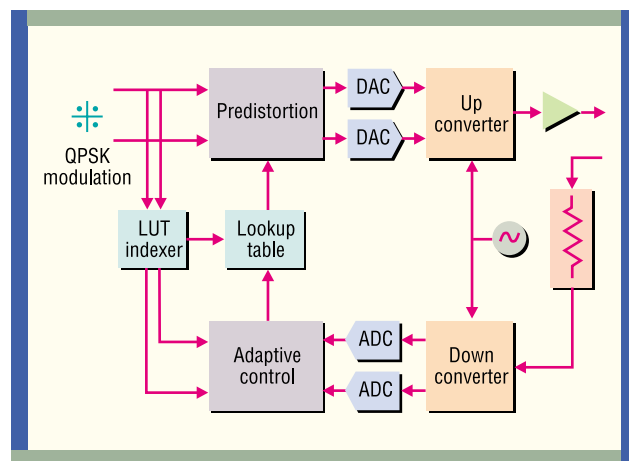


Figure 4. DSP-based adaptive baseband predistortion can greatly simplify the design of the RF power amplifier and enhance its efficiency. The solution shown here uses feedback from the PA output to update a lookup table operating on the baseband samples.

conversion processing in software, using high-speed DSPs or user-programmable gate arrays. Some companies offer FPGAs suited to that task, and TI and others offer DSPs with sufficient processing speed, such as the TMS320C6000 and C5000 DSP platforms. Additionally, numerous third-party suppliers, including TI Third Party Network Members, are providing custom algorithms or development tools.

Digital signal processing engines. When the IF signal is in complex baseband form, you can process the signals using a dedicated ASIC (very limited flexibility), an off-the-shelf DSP solution (or a DSP core), or one or more FPGAs. Again, the maximum flexibility sought in our software-programmable WLAN solution is achieved using the DSP or FPGA option. A growing number of high-speed DSP devices can handle data rates of several megabits per second, including the C5000 platform of ultralow-power devices for portable use and the C6000 platform for very fast applications. ◆

REFERENCES

Bateman, Andrew, *Digital Communications: Design for the Real World*, Addison-Wesley, London, 1998.

Bateman, Andrew, and Iain Paterson-Stephens, *The DSP Handbook*, Prentice-Hall, Upper Saddle River, N.J., forthcoming.

Kenington, Peter B., *High Linearity RF Amplifier Design*, Artech House, Boston and London, 2000.

Andrew Bateman is the CEO of Avren Ltd., a design and consultancy company that advises communications and IC companies on technologies, standards, and partnerships for next-generation wireless systems, and of DSPStore.com. Previously, he was a professor of communications and signal processing at Bristol (U.K.) University and cofounded Wireless Systems International Ltd., where he was the business development director. He is also the author of three books on digital communications and digital signal processing.

MP4110 MPEG-4 Simple Visual Profile Codec
MP4010 Multimedia Application Framework

A high-performance, device-optimized, software-programmable, eXpressDSP™ Compliant **MPEG-4** solution optimized for the Texas Instruments TMS320C6000 family of Digital Signal Processors.

The **MP4110** MPEG-4 Simple Visual Profile Codec provides dynamically configurable real-time video compression/decompression ideal for numerous multimedia applications including video servers, multimedia gateways, headset multimedia systems, surveillance systems, network-connected cameras, and consumer electronics.

The **MP4010** DSP Multimedia Application Framework establishes a scalable software foundation for the rapid integration of numerous ingenious multimedia algorithms, as well as third party algorithms.

DSP solutions enabling next-generation multimedia applications. Now that's Ingenient.

ingenient technologies
<http://www.ingenient.com> • 947.357.1960 • 947.357.1961 (fax)

DSP
 TI&S NETWORK

MESi is unlocking the barriers to widespread DSP software modem use.

Are you a specifying engineer or an engineering manager? Do you need embedded modem software products for network equipment, set-top box, mobile medical or remote data loggers? Are you finding the barriers to widespread DSP software modem use locked up tight? MESi can get you component and system software for Texas Instruments DSP devices "barrier free" with:

- **Low** MP's and memory
- **Easy** user interface
- **Simple** royalty-free licenses
- **Best** published prices

Just log on to www.mesi.net for the DSP software you need. Check out our products and prices, then E-mail us at info@mesinet.net. We'll get back to you right away. Barriers, you say? What barriers?

MESi
 Miller Engineering Services, Inc.
www.mesi.net

DSP
 TI&S NETWORK

In just a few short phases, you can optimize C-coded reference modems to meet higher-density targets.

Optimizing Modems Using Code Composer Studio and TI Resources

By Ghassan Farah

For many of the general telephony stream-processing tasks, the C optimizer for the Texas Instruments TMS320C6000 DSP platform can yield higher densities with no assembly coding. Other technologies require a healthy dose of optimization to reach target densities.

You can take steps to optimize C-coded reference modems to meet higher-density targets. How high? C-baseline modems, for example, can soar from 6 per 200-MHz C6201 to 28 in four short project phases. A fifth project phase can take the number of channels to 48 per DSP.

In fact, for the MSP MEDIA Gateway line of DSP resource boards based on C6000 DSPs, Commetrex undertook the four phases, and the process worked. Our MSP-320 PCI board, with two C6201 DSPs and a quad E1/T1 network interface, needed 48 to 60 channels of processing from each DSP. For many of the general telephony stream-processing tasks, the C6000 C optimizer gave us the densities we needed with no assembly coding.

"Out-of-the-box" C-coded modems, which are a reference design

and written for understandability rather than efficiency, might compile to, say, six simultaneous modems. You should be able to double that by guiding the modems through the Code Composer Studio (CCS) optimizer and by ensuring that your memory layout takes advantage of the C6000's on-chip RAM.

CCS includes an optimization tutorial that provides a recommended code development flow consisting of four phases (Figure 1). (A similar tutorial is in the *TMS320C6000 Programmer's Guide*.)

Phase 1 involves compiling and profiling your baseline C code. Before you begin any optimization effort, use the profiling tools to identify the performance-critical areas in your code.

Phase 2 involves compiling with the appropriate optimization options and analyzing the feedback provided by the compiler to improve the performance of your code.

Phase 3 is a critical phase during which you use a number of techniques to tune your C code for better performance.

Phase 4 is needed if the performance of certain areas of your code must be improved beyond the tuning phase. After yet another profile of the code, you can extract the performance-critical areas and rewrite them in linear assembly language.

THE FIRST THREE PHASES

Phase 1 establishes your baseline. You have a goal—for example, your system requirement might be a statistical mix of 48 modems on one

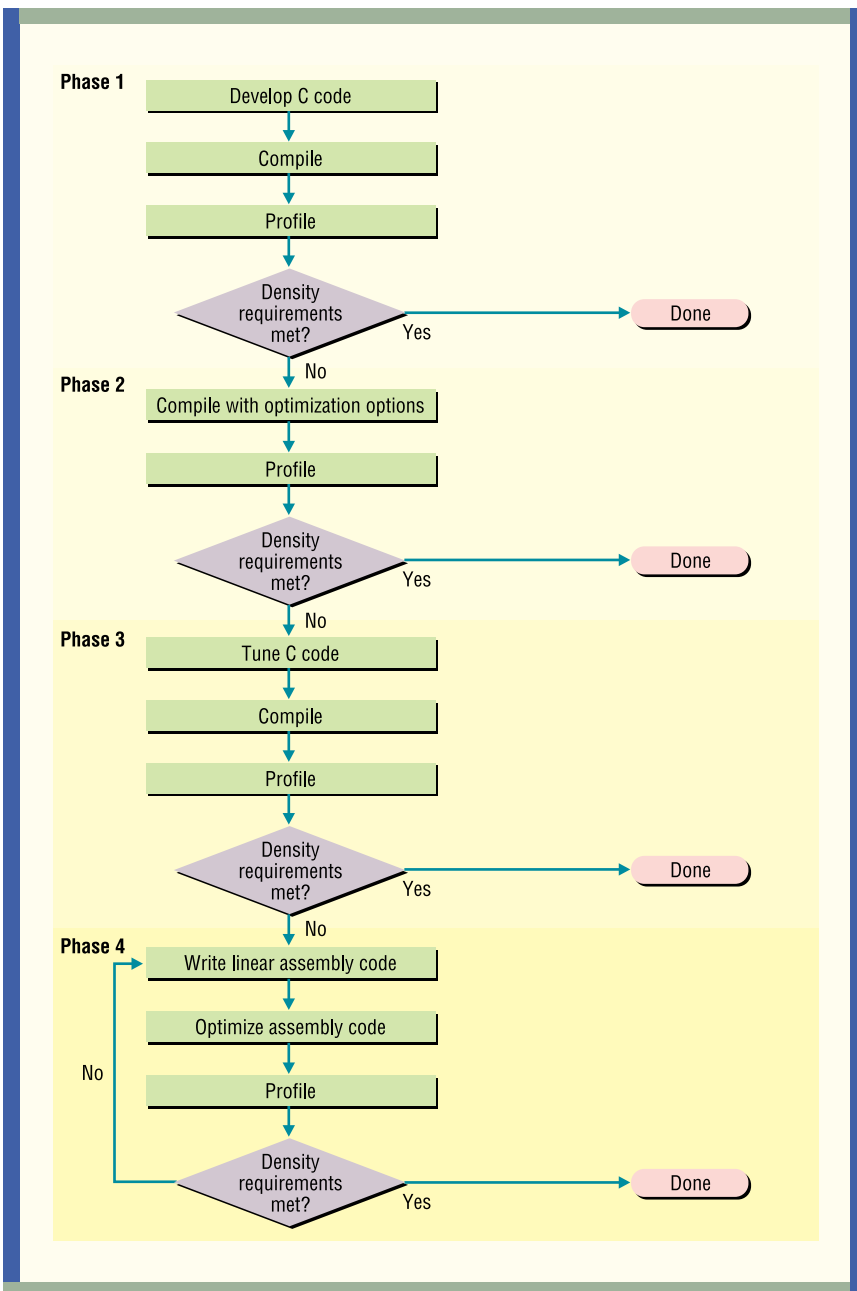


Figure 1. The Code Composer Studio optimization tutorial recommends a code development flow consisting of four phases. The first three phases focus on utilizing the optimization abilities of the TMS320C6000 compiler to achieve high code performance while maintaining the code in C. The last phase involves linear assembly coding of those portions of the code whose performance needs to be improved further. (This figure is based on the one on p. 1-4 of the TMS320C6000 Programmer's Guide).

200-MHz C6201. Always maintain the C-coded baseline as your reference code. Because it's often developed very straightforwardly, leaving it as a reference will be valuable if you have to diagnose a problem. Make your improvements there, then factor them into the optimized version to produce a bit-exact version.

Phase 2 involves compiling using the optimization options. The optimizer, combined with a judicious memory layout, can more than double the number of modems on one chip. Allocate a few weeks for the effort. But note that the optimizer is capable of "breaking" the modems in a few places, so you may have to modify some pieces of the C code. Also, you may find that some of the changes you make to the C code to improve the optimizer's results when using CCS 1.1 aren't required when using the release 1.2 optimizer; therefore move to 1.2 if you can.

In phase 3, you tune the C code. There are a number of techniques to refine your C code and greatly increase its efficiency. The goal is to allow the compiler to schedule as many instructions as possible in parallel, especially for MIPS-intensive loops, by providing information concerning the dependencies between instructions. You can use certain key words that give the compiler hints as it tries to determine dependencies.

Another useful technique in the tuning phase is to use intrinsics, which are special functions that map directly to C6000 assembly instructions. These functions are usually not easily expressed in C. They allow you to have more precise control over the selection of instructions by the compiler.

For example, some intrinsics operate on data stored in the low and high portions of a 32-bit register. Consequently, if you're operating on a stream of 16-bit values, you

Listing 1: Fixed-Point FIR Filter with Data Move

```

/*****
* Routine Name: FIR_Filter_Shift
*
* Description:
*   Performs fixed-point implementation of FIR filtering with
*   data move.
*
* Calling Sequence:
*
* short FIR_Filter_Shift(          short *taps,
*                               short *coefs,
*                               unsigned short length,
*                               unsigned short base)
*
* Where:
*   taps   = pointer to filter taps delay line
*   coefs  = pointer to filter coefficients, which are
*           stored in reverse order
*   length = length of taps delay line
*   base   = base of filter coefficients
*
* Returns:
*   A filtered sample
*****/

short FIR_Filter_Shift(          short *taps,
                               short *coefs,
                               unsigned short length,
                               unsigned short base)
{
    int sum = 0;

    taps += length - 1;

    while (length--)
    {
        sum += *taps * *coefs++;
        *taps-- = *(taps-1);
    }

    /* round and remove base */
    return( ( sum + (1<<(base-1)) ) >> base );
}

```

can use word (32-bit) accesses to read and process two 16-bit values at a time.

Even though phases 2 and 3 may double the number of simultaneous instances of the code running on one chip, the modems are still coded in C that's easy to understand and maintain.

PHASE 4: CIRCULAR ADDRESSING

At this point, if your performance requirements are not yet met, you go on to phase 4: converting MIPS-intensive portions of the code into linear assembly code. This form of assembly code doesn't require that you provide functional unit selec-

tion, pipelining, parallelization, or register allocation; those tasks will still be performed by the compiler. It will, however, give you more control over the exact C6000 instructions to be used. You can also pass more useful information to the tools, such as which memory bank is to be used.

Modems use a number of delay lines for the different filters, resulting in MIPS-intensive memory shifting. You can avoid that by employing the circular addressing feature of the C6000 in your linear assembly code. It's not unreasonable to set a goal of doubling the number of modems from 12 to 24 in this step alone.

For the most part, a modem is a series of filters. Each filter is computed from a sequence of input data, or taps, and an equal number of coefficients. A multiply-accumulate operation is performed with each tap and a corresponding coefficient. After the computation, the taps are shifted to make room for the new input (Figure 2a). Circular addressing changes the starting point for the MAC cycle, eliminating the shifting altogether (Figure 2b).

Without hardware support for this operation, the C code for the iterative loop is of the form in Listing 1. The C6000 has hardware support for circular addressing, though. By setting the addressing mode register (AMR) appropriately, you can specify the general-purpose register or registers that will be used for circular addressing, as well as the size of the memory block that will be addressed circularly (Listing 2).

Just as using the optimizer has its challenges, so can adding circular addressing. You might find that you add circular addressing and then the optimizer breaks it. It turns out the optimizers in both CCS 1.1 and 1.2 don't take circular addressing into account. For example, the optimizer will often move an address from a register configured for circular

addressing to another register before performing address manipulations.

When using the optimizer with circular addressing, you might have to experiment with a number of alternative codings to arrive at a solution that the optimizer respects. (The new Code Composer Studio 2.0 from TI supports circular addressing directly from C code.)

You should see a significant improvement with circular addressing. Take our V.29 receiver (9,600 b/s) as an example: After the first three phases of our project, it consumed 222,188 cycles for each 10 ms of PCM data (80 samples). By modifying just the first two sections—the pulse-shaping and Hilbert filters—for circular addressing, we brought that down to 185,759 cycles. Changing the interpolating and baud timing recovery filters to the circular addressing mode reduced it to 155,677. Finally, changing the adaptive filtering and update routines shrank the cycle count down to 101,429—a reduction of better than 55 percent. (For a more in-depth discussion of circular addressing on the C6000, refer to the TI Application Report *Circular Buffering on TMS320C6000* [SPRA-645.PDF].)

Since a V.17 receiver (14,400 b/s) is essentially the same code as the V.29 receiver but executes from different tables, these changes cause similar reductions to the V.17 receiver. However, we still need to optimize the Viterbi decoder. (Of the three common modems used to transfer fax-image data, only the V.17 modem uses Viterbi decoding.)

Trellis coding is a forward error correction scheme that reduces a modem's bit-error rate for a given amount of channel noise by adding certain redundant information to the channel. The information reduces the chance that noise will create data errors, in effect increas-

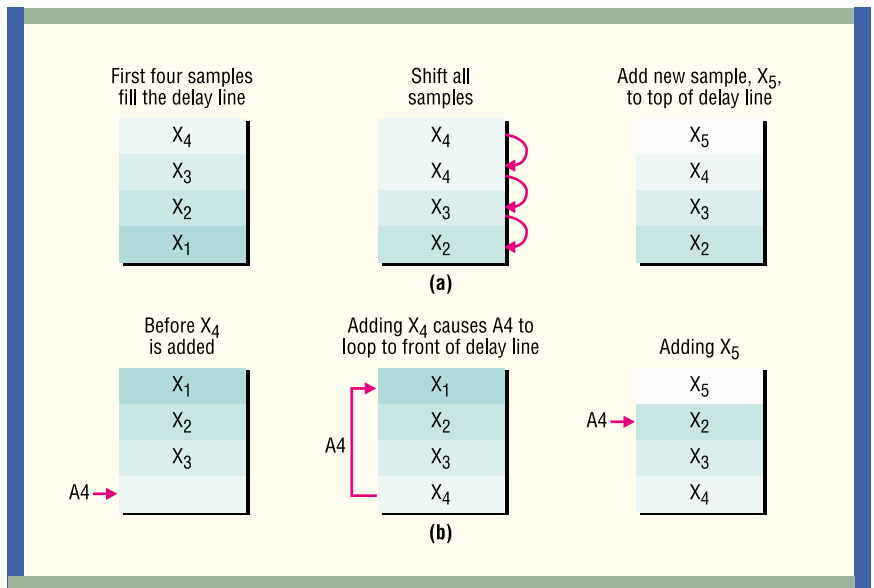


Figure 2. A new sample, x_5 , is added to the delay line of a four-tap filter (x_1 is the oldest sample in time) using the sample-shifting method. Three shifts are needed before x_5 is placed at the top of the delay line (a). Using circular addressing, register A4 (set up to be used in circular mode) automatically wraps back to the beginning of the delay line after x_4 is added and the end of the delay line is reached. When x_5 is added, it overwrites the oldest available sample, x_1 , thereby eliminating the shifting altogether (b).

ing the distance between code points. The Viterbi decoder decodes the Trellis sequence and determines the most likely set of transmitted points. However, it's expensive in terms of MIPS. C-coded Viterbi decoder alone took 140 percent of the cycles that the entire V.29 receiver took. In other words, the V.17 receiver was 2.4 times as expensive as the V.29.

Fortunately, help is available on the TI Web site (www.ti.com). When you download *Implementing V.32bis Viterbi Decoding on the TMS320C6200 DSP* (SPRA-444.PDF), you'll find the decoder in very tight assembly code. You can't just drop it in, though. You'll have to adapt it to your environment.

To make the decoder reentrant, change global variables to per-channel contexts and watch for bugs. You

should achieve spectacular results: A straight C-coded Viterbi consumes approximately 150,000 cycles for 80 samples. Substituting TI's assembly code takes that down to an incredible 8,000 cycles. Our V.29 receiver is now 101,429 cycles, and the V.17 receiver only 108,840—and we haven't begun to "vectorize."

Using a statistical mix of modems yields 28 simultaneous channels. In worst-case nonblocking terms, that's 18 simultaneous V.17 receivers. You should receive similar results for similar algorithms by using the optimizer and circular addressing.

BEYOND PHASE 4: 'VECTORIZE'

If you still haven't reached your performance requirements, you

Listing 2: FIR Filter Using Circular Addressing with Hardware Support

```

; Replacement for FIR_Filter_Shift
; PARAMETERS:
;   A4 - (in) *coefs
;   B4 - (in) *taps (base address of circular buffer)
;   A6 - (in) length (length of delay line)
;   B6 - (in) block_size
;   A8 - (in) write_offset (where the next value will be written)
;   B8 - (in) base

.global _FIR_Filter

_FIR_Filter .cproc coef_block,taps,A6,B6,A8,base

        .reg    old_amr,ar,coef,sum1,d1,one,round,offs

; coef_block = coefs + ((length-1) * sizeof(short))
SUB     A6,1,d1                ; d1 = length - 1
SHL     d1,1,d1                ; d1 = (length-1) * sizeof(short)
ADD     coef_block, d1,coef_block ; coef_block = coefs + d1

SHL     B6,16,B6               ; set block size of circular buffer
SET     B6,8,8,B6              ; set B4 to circular mode
MVC     AMR,old_amr            ; old_amr = AMR
MVC     B6,AMR                 ; AMR = addressing_mode

ZERO    A1                     ; sum = 0

; ar + write_offset is where we will write the next sample
; advance to the most recent word (write_offset - 2)
; ar = taps + (write_offset - 2)
SUB     A8,2,offs
ADDAB   taps,offs,taps
MV      A6,d1

startloop: .trip 1
LDH     *taps--,ar
LDH     *coef_block--,coef
MPY     ar,coef,sum1
ADD     sum1,A1,A1
SUB     d1,1,d1
[d1] B   startloop

; round and remove base
; sum = ( sum + (1<<(base-1)) ) >> base
SUB     base,1,round
MVK     1,one
SHL     one,round,round
ADD     round,A1,A1
SHR     A1,base,A1

MVC     old_amr,AMR            ; restore original addressing mode

.return A1
.endproc

```

might consider going on to phase 4: changing the flow of data through your code to reduce function calls and utilize more loops that can be optimized easily. For modems, one approach to accomplish that is to “vectorize” the algorithm’s implementation.

The sample rate section of the receiver consists of the following components in series: the pulse-shaping filter, the Hilbert transformer, the demodulator, and the interpolator. Without vectorization, the sample rate section of the receiver processes one sample at a time, taking it through each successive section. Consequently, the overhead of calling each filter in the sample rate section is incurred 80 times for each 80-sample buffer. With vectorization, the sample rate section is called once for each 80-sample buffer. An input buffer of 80 samples is then passed to the pulse-shaping filter, which produces 80 samples to be passed to the Hilbert filter, which in turn produces 80 outputs, and so on. In the sample rate section, the number of function calls required to process 80 samples is reduced from 320 to just 4. In addition, processing the input buffer in a loop format as opposed to sample by sample allows the optimizer to do a better job of pipelining, significantly improving efficiency.

We haven’t completed the vectorization phase of this project, but we will report the results on our Web site (www.commetrex.com) when we do. ◆

Ghassan Farah (Ghassan_Farah@commetrex.com) is manager, signal processing technologies, at Commetrex Corporation in Norcross, Ga. He has four years’ experience in designing and implementing a variety of DSP algorithms. His technical interests include data and fax modems, telephony, speech coding, and signal classification.

TCP/IP *for*

TI DSPs

Delivering precisely
what you need.

Since 1994, Precise has been a leading
supplier of embedded Internet protocols on
Texas Instruments' TMS320™ DSP Family.

TMS320C3x™

1994

TMS320C4x™

1995

TMS320C62x™

1998

TMS320C67x™

1999

TMS320C54x™

1999



Precise Software Technologies Inc.
301 Moodie Drive, Suite 308
Nepean, Ontario
Canada K2H 9C4
Sales: 800-265-9833
Phone: 613-596-2251
Fax: 613-596-6713
E-mail: info@psti.com



www.psti.com

New software tools are taking aim at multiprocessor DSP systems, particularly for the C6000 DSP platform.

Speeding the Development of Multi-DSP Applications

By Fiona Culloch

Although many of the latest top-of-the-line processors are extremely fast—and the 1.1-GHz TMS320C6000 from Texas Instruments is extremely fast—some customers still have performance requirements that mandate a multiprocessor solution. That's why many multiprocessor C6000 systems are already in the field.

New software tools like TI's Code Composer Studio (CCS) are helping to speed software development, but for the most part they're targeted at the DSP device. Although CCS supports loading and debugging multiprocessor target systems via JTAG, its focus is understandably the DSP chips, not the additional board-specific hardware involved in inter-processor communications.

Complementary software tools are needed to truly unlock the promise of faster software development for multiprocessor DSP systems. Indeed, software tools for multiprocessor C6000 development are now emerging, such as Diamond from 3L, Virtuoso from Eonic

Systems, Pegasus from Jovian Systems, and Accelera from Spectrum Signal Processing (which is specific to the company's boards).

Diamond and Virtuoso ease the software burden of coordinating many processors with features like Eonic's Virtual Single Processor model and 3L's "virtual channels." These features free you from the protocol complexities of forwarding messages among a network of processors and the interface details of your particular communications hardware. This class of products also provides ready-made software that lets a range of off-the-shelf DSP development boards communicate with user-written GUI code on a host PC.

Pegasus and Accelera are higher-level tools: graphical development environments for multi-DSP applications. Instead of writing C code to implement DSP algorithms and interprocessor communications, you can simply choose from a palette of predefined functional blocks (or add your own) and visual-

ly place the blocks onto processors.

Both kinds of tool are useful, since you often want to work at both levels. In fact, Pegasus can use either Diamond or Virtuoso as its underlying communications framework. Accelera uses Spectrum's own quicComm software as its communications layer, as it targets proprietary communications hardware.

COMMON PROBLEMS WITH MULTIPROCESSORS

Ever since computers became inexpensive enough to use more than one on the same problem, the industry has gained a lot of experience about the significant software problems introduced by multiprocessing.

Partitioning of the problem and its data. Getting multiple processors to work together effectively on the same task involves splitting the job into chunks that can be efficiently processed independently or with minimum communications. Except in special cases, partitioning is as much art as science. Creative engi-

neering is needed here, and software is of little help.

Processor communications and synchronization. Although some systems can consist of multiple, completely independent activities, in most cases the processors must exchange data via shared memory or point-to-point links.

With shared memory, each processor has access to a shared bank of memory, often through a common bus. The processors signal the availability of data by interrupts. Point-to-point links provide a dedicated hardware path between each pair of processors that need to communicate. Some DSP hardware directly supports that model, such as TMS320C4x communications ports (comports) or C6000 multi-channel buffered serial ports (McBSP).

Simulation. It's hard to develop the software for a multiprocessor application on custom hardware before the hardware is available and substantially debugged. The usual single-processor solution—developing and testing software on a host platform, then porting it to the target system—isn't feasible if the software must assume the presence of the target hardware's interprocessor communications features.

Debugging. The only practical way to debug some common failure modes of multiprocessor systems, especially deadlocks caused by a particular pattern or ordering of communications, is to log the interprocessor communications (rather than single-stepping code, which is next to useless for some problems). As mentioned, that can require sophisticated software support in a system built from point-to-point links.

The key element for solving communications, simulation, and debugging problems is *abstraction* of interprocessor communications and

synchronization. Unfortunately, the opposite happens in most DSP projects. Instead of keeping the details of the communications hardware out of the application code, developers tend to tie them as closely together as possible, in a misguided quest for "efficiency." Consider some concrete examples.

The SMT3xx range of modular C6000 DSP hardware from Sundance Multiprocessor Technology is

different for different modules in the range), and field the interrupt generated by the DMA channel on completion of the transfer.

Although the code isn't particularly complex, it's usually slow to write. Not only do you have to completely absorb all the low-level details of the hardware before writing any code, but you must also account for "kinks" in the hardware that may not have been fully documented.

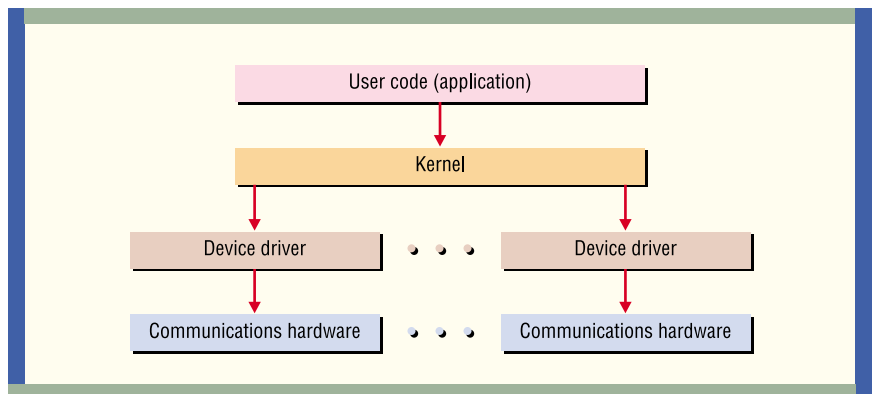


Figure 1. Device driver software provides plug-in knowledge of the specific hardware and maps between a common, device-neutral API and actual interprocessor communications hardware, which may differ for each DSP.

based on point-to-point Sundance Digital Links (SDLs) implemented by an on-board FPGA. There are various implementations of the technology: 20-Mb/s versions backward-compatible with TI's TMS320C4x TIM-40 module comport standard, as well as faster ones that take advantage of more recent technologies. From a software point of view, to send data over a link, code must correctly initialize both the FPGA and the C6000 External Memory Interface (EMIF) CE1 control register, assign a CPU interrupt line to the transfer (avoiding any in use by concurrent I/O operations), initialize a C6000 DMA channel to transfer the data between memory and the FPGA data port (the addresses of the FPGA control registers, and the bits within them, are

That's a simple case compared with Spectrum Signal Processing's Daytona and Barcelona (dual and quad) C6000 boards, which use shared PCI SRAM blocks for communications. The equivalent code to send a message from one processor to another must correctly initialize the dedicated communications ASIC (Hurricane), which implements a point-to-point link and has 64 individual control registers, as well as shared-memory buffer data to ensure that the buffer addresses are valid in the address space of each processor. It must chop the data to be transferred into chunks that fit into the SRAM bank visible to the Hurricane ASIC and, for each chunk, create a DMA channel control program in memory to drive Hurricane and start it. last, it must

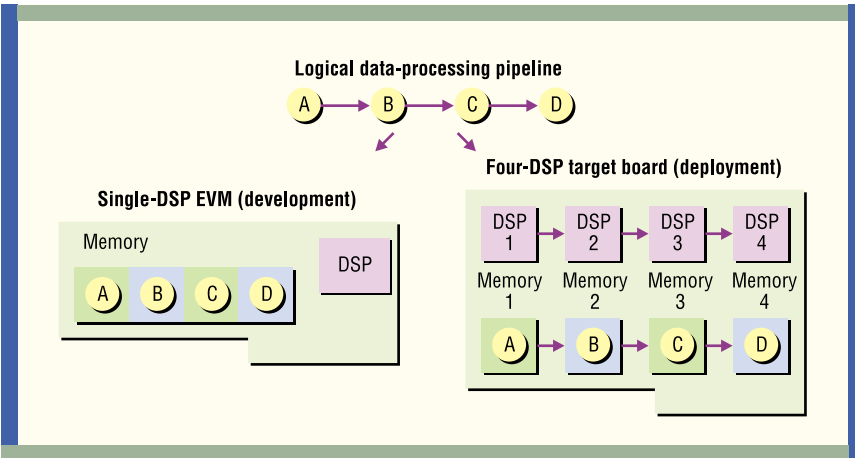


Figure 2. A four-stage pipeline can be developed on a single-processor board, then deployed without source changes on a four-DSP board for a 4x speedup.

field interrupts from the ASIC when channel program operation is com-

plete, either moving on to the next chunk (which may involve copying

further data into the Hurricane SRAM) or signaling the user code that the transfer is completed.

Although the managing code is often scattered throughout a project's application-level programs, it's obvious that the same abstract operation is performed in both cases: Send this much data from here in local memory to a receiver on another processor.

Abstraction decouples the application software—the DSP algorithms—from the communications hardware and contrasts with the point solutions used on many multiprocessor projects, where you try to tackle interprocessor communications by directly manipulating the underlying hardware in the application code. Although that approach

Are you a Control Freak?

Combine our C6x DSP baseboard and Servo16 OMNIBUS Module for wide-channel-count servo control applications

Features

- ▶ Dedicated 160 MHz, floating-point DSP (PCI plug-in & Stand Alone versions)
- ▶ 16 Channels - 100 kHz - 16-bit A/D & D/A
- ▶ Comprehensive C/C++ cross development tools & servo algorithm templates
- ▶ Windows 9x/NT/2000 drivers

Control

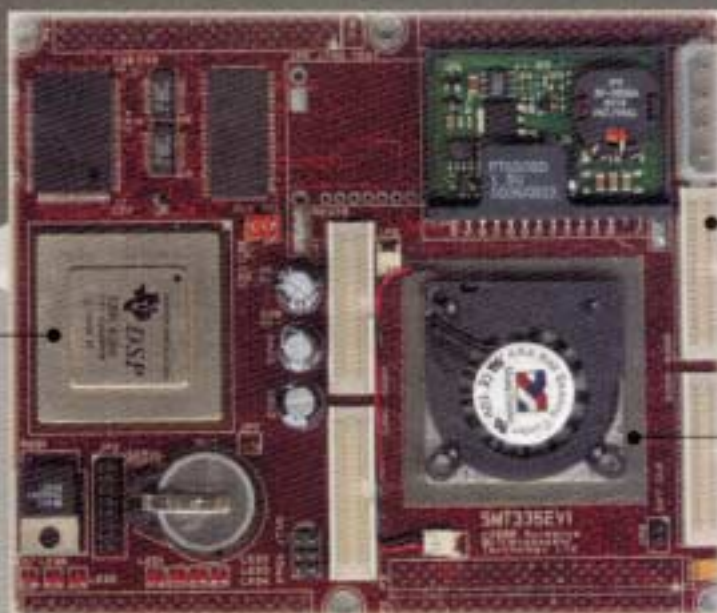
- ▶ Motors
- ▶ Piezo Actuators
- ▶ Other analog transducers

Innovative Integration
www.innovative-dsp.com
805.526.1100 phone • 805.526.1100 fax



Hand-In-Hand Power of DSP + Flexibility of FPGA

SMT335E



TI's C6xxx DSP

SDB Connector

Xilinx Virtex 2000E FPGA

Features: Compact module, four SDB interfaces for fast IO or for inter processor communication(200MB/S/SDB), six built-in comports, FLASH for embedded applications. Available support for PCI, CPCI, VME and VXI. Can cascade multiple modules.

SUNDANCE DIGITAL SIGNAL PROCESSING INC.

Tel: (775) 827-3103 USA

SUNDANCE MULTIPROCESSOR TECHNOLOGY LTD.

Tel: +44 (0)1494 793167 UK

Email: sales@sundance.com <http://www.sundance.com>

SUNDANCE

works after a fashion, in most projects time inevitably limits the scope of the sophisticated code required to assist development. Support for tasks like debugging I/O from a node to a host system and the hardware independence required for simulation are omitted in the rush to get something working. Ironically, lacking simulation facilities, the project is likely to take much longer than if you used a tool that supports hardware-independent communications. Such tools offer many benefits.

Complete, off-the-shelf solutions. Solutions for the communications, simulation, and debugging problems are feasible if the project is able to take advantage of commercial off-the-shelf (COTS) multiprocessor hard-

ware in conjunction with software tools that abstract interprocessor communications. For example, Diamond software runs out of the box with multiprocessor C6000 boards from Spectrum Signal Processing and Sundance Multiprocessor Technology; it includes drivers for the different interprocessor communications hardware used by the boards, removing a requirement for scarce driver development skills from the project's critical path.

Simple communications API. High-level tools for multiprocessor software development solve communications problems by providing a clean, simple API for application code.

Software development for cus-

tom hardware using COTS boards.

When you use tools that decouple application code from the communications hardware, you can develop software with COTS hardware, even if the final target uses custom multiprocessor boards. Good multiprocessor development tools let you port the working code to the target system without application source changes, thus addressing the simulation

problem, provided that multiprocessor COTS hardware is available.

Multiprocessor software development on a single-processor platform. Being able to develop multiprocessor software on a single-processor platform means that not only is the application code independent of the communications hardware that connects the processors, it also can be independent of the number of processors and their connectivity. In particular, you can take a multiprocessor application consisting of separate programs for several processors and run it unchanged on a single processor, as long as there is adequate memory. The software automatically relocates each program to a separate position in the single processor's memory. An RTOS kernel time-shares the processor between the programs and transforms what were previously interprocessor communications calls to the kernel.

More interestingly, it's equally possible to go the other way: Develop a system of independent programs (processes) on one processor—for example, a simple evaluation module (EVM) from a DSP silicon vendor—and then later distribute the processes to separate DSPs when real target hardware appears, again with no application software changes (Figure 1). That feature widens the range of options for solving the simulation problem. The same flexibility that supports switching between single-processor and multiprocessor configurations without recoding also supports boosting the performance of properly designed applications by simply adding more processors, again without code changes.

Development under Windows. A corollary of independence from the underlying communications hardware is that, with software processes in a host environment like Windows

MP3

imagine
TECHNOLOGY, LLC

As an official third party vendor of Texas Instruments, Imagine Technology offers tested and approved algorithms for a wide range of DSP based processors, which are eXpressDSP™ compliant.

Available Algorithms

- Convolutional Coder
- DES & Triple DES
- DTMF Generator/Decoder
- V.22 Modem
- Call Progress Detection
- BPSK Modem
- G.726 Encoder/Decoder

The MP3 algorithm provides MPEG-1 and MPEG-2 audio decoding for layer 3 streams and supports all standard sampling rates for the following Texas Instruments DSP generations:

- TMS320C54x™ DSP
- TMS320C55x™ DSP
- TMS320C62x™ DSP
- TMS320C64x™ DSP
- TMS320C67x™ DSP

Imagine Technology, LLC
1331 SW 84th Street
Lincoln, Nebraska 68532 USA
Tel. 402-438-0589 Fax. 208-545-7811
www.imagine-technology.net

eXpressDSP Compliant

DSP
TEXAS INSTRUMENTS

NT, it's possible to simulate multiple processors during development, expanding your armory of techniques for developing working multiprocessor software in parallel with custom hardware development (simulation problem). The trade-off is the increased availability of software tools on the host system versus difficulties introduced by simulating the target system on a CPU with a different architecture (assembly functions can't be directly tested, for example).

Framework for multiprocessor development. As you can see, you must watch out for a number of pitfalls in multiprocessor systems before achieving application speedup. Most high-level multiprocessor development tools help by providing a

ready-made working framework for multiprocessor application design. The abstract model of several such tools, including Diamond, is based on C. A. R. Hoare's *Communicating Sequential Processes* (Prentice-Hall, Englewood Cliffs, N.J., 1985). Software based on communicating sequential processes (CSP) has been widely employed in the industry for at least ten years. Using such a road-tested model in place of ad hoc twiddling of hardware control registers eliminates many design and implementation errors and helps with communications and debugging.

The abstraction that produces the benefits is almost entirely conceptual—you have no implementation overhead above that of the traditional hardware-specific approach,

other than selecting the required hardware-specific implementation of each communications function with a pointer rather than directly. The extra memory load required is orders of magnitude less than the amount of time required for the communications itself.

AN IMPLEMENTATION

How is the framework concept presented to you, the application developer? We'll focus on frameworks based on writing C code, using 3L's Diamond as an example, but most of the underlying concepts also apply to other high-level multiprocessor development frameworks. Note that a graphical tool like Pegasus essentially acts as a "wizard" to generate

SRS Labs' Voice Intelligibility Processor (VIP™)

Hear the difference

- SRS Labs' VIP algorithm dramatically improves the quality and intelligibility of speech on:
Traditional and cellular phones, VoIP call center equipment, headsets, microphones, digital radios, televisions and broadcast equipment
- VIP provides significant improvements in the clarity and quality of synthesized voice in text-to-speech and voice portal applications
- VIP selectively enhances speech formants with minimal increase to overall SPL (sound pressure level)
- VIP is eXpressDSP™ compliant on the TI 54x and 55x DSP platforms

POWERED BY
DSP
THE ARCHITECT
"SRS Labs' VIP has a small footprint so it can easily work in conjunction with other 3rd Party Network members' software applications."
— Dennis Barrett, Texas Instruments

Contact us to see how VIP™ can improve your product
Call 1-800-243-2733 or email sales@srslabs.com

SRS  **VIP**
www.srslabs.com by **SRS** 

©2001 SRS Labs, Inc. All rights reserved. SRS, VIP and the SRS symbol are trademarks of SRS Labs, Inc.

VIP dramatically improves the quality and intelligibility of speech especially in noisy environments.

What?

or



C code from a visual block diagram, but it still allows you to work with the generated code at that level.

The key to most of the framework benefits is the hardware-independent API for interprocessor communications. Some multiprocessing aspects of Diamond and Virtuoso were influenced by Occam, the native language of the Inmos transputer, which was itself an implementation of Hoare's CSP concept. The CSP notation expresses concurrency by mathematical operators denoting synchronized message passing. Like other languages based on the CSP model, Occam has a special syntax for transmitting messages (! to send, ? to receive).

The key to most of the framework benefits is the hardware-independent API for interprocessor communications.

However, for a CSP-based concurrency framework to operate successfully with standard C tools like CCS, that syntax must give way to function calls. In Diamond's case, the API is basically two functions that operate on abstract, point-to-point channels, represented by the CHAN data type:

```
void chan_in_message(int length, void
    *buffer, CHAN *channel);
void chan_out_message(int length, void
    *buffer, CHAN *channel);
```

Those primitive operations handle both message-based communications and (implicitly) synchronization: After calling `chan_out_message`, the sending thread is blocked until the message is received. Similarly, a thread that calls `chan_in_message` is suspended until the incoming data arrives on the specified channel. Virtuoso generally

uses channels directly only for system processes; its application-level tasks use mailbox and FIFO objects that the system constructed from channels. But with those frameworks, calling a simple function encapsulates all the hardware-specific work required on a particular board—for example, on the Sundance SMT3xx modules, setting up and driving the FPGA, fielding the interrupts it generates, and controlling a C6000 DMA channel (and its interrupts) to move the data between memory and the FPGA that handles interprocessor I/O.

A code-based framework presents its hardware-independent communications services as API functions

in the usual way, with a header file and corresponding run-time library. C code using the API is compiled and linked by CCS's tools, as usual. Thus, just as the CCS optimizing C compiler frees you from explicitly coding DSP instructions, a code-based framework frees you from coding board-level interprocessor communications at the hardware register level.

Frameworks differ significantly in the area of system configuration. Generally, a framework has a configuration file that governs the mapping of software tasks onto the physical topology of the processor network. Code-based systems like Diamond or Virtuoso use a text file; graphical systems like Pegasus generate the file automatically from the system block diagram on screen.

The underlying configuration technology also varies. Virtuoso

uses the standard linker to bind tasks, producing one executable file for each node so that tasks on the same node share a single namespace for external variables and functions. Diamond, on the other hand, can bind multiple separately linked images for each node into a single bootable file for the whole network. Any node can run multiple programs, each with its own namespace, like a process in Unix or Windows NT. An RTOS kernel component with full preemptive scheduling of dynamically created threads is loaded onto each target processor. Basic debugging takes place via JTAG with CCS, as usual. In fact, you can view a communications framework like Diamond as complementary to CCS, extending it with mechanisms for simplified handling of multiprocessor applications that have significant interprocessor communications requirements.

As well as sending and receiving messages, the other primitive required by systems based on the CSP model is waiting until a message arrives over one of several alternative channels, where the source of the message is not known in advance (similar to the select operation in some Unix variants):

```
i = alt_wait(n, &chan_1, &chan_2, ...,
    &chan_n);
```

The communications channels can be implemented with any hardware that allows for some form of communications and synchronization. For example, 3L previously implemented the same API on bit-serial transputer links, TMS320C4x comports, C6000s using PCI-bus shared memory and interrupts for signaling, and C6000s using VMEbus shared memory and interrupts. Applications can run unchanged on all of them.

What do you do to take advantage

of the leverage provided by that framework? With supported COTS multiprocessor hardware—nothing. The implementation of the framework's abstract communications primitives in terms of the supported hardware is built into the system. For other custom hardware, you create a link device driver for the framework.

DRIVERS FOR CUSTOM HARDWARE

To make use of a high-level framework for multiprocessor application development on custom hardware, you must write a device driver that maps from the framework's abstract communications API to the available hardware (Figure 2). Three approaches are possible: porting kits,

fee-based service, and IP licensing.

With the first approach, you write the driver based on a porting kit provided by the framework vendor that documents the interfaces between the driver and the rest of the system. The porting kit must contain sufficient software components so that you can link your custom drivers into the system.

With the second, you go to the framework vendor. In most cases, the vendor has a great deal of experience writing drivers that interface their software to the COTS boards that it supports and is willing to write custom drivers for a fee.

IP licensing is an intermediate approach to speed up driver development based on a porting kit. You license working source code for

existing implementations from the framework vendor and use that code as a base for development.

In all cases, as soon as a working driver exists for the custom hardware, multiprocessor code written with the framework—either on a supported COTS board, on a single-processor EVM, or on Windows—should run unchanged on the new hardware. ♦

Fiona Culloch (fc@3L.com) is technical support director at 3L Limited in Edinburgh, Scotland, where she has been involved with real-time kernel development and software to simplify integration of DSPs with host GUI applications. From 1985 to 1987, she was software development manager for the compilers group at Lattice Logic Limited.

SRS Labs' WOW™ Technology

Achieve BIG sound from small speakers or headphones

- SRS Labs' WOW algorithm significantly improves the quality of digitally compressed stereo audio files
- WOW produces a much wider and taller sound image field and deep rich bass
- WOW technology is perfectly suited for many popular products such as: Laptop computers, MP3 players, personal/portable audio products, Internet appliances, PDA's, television, mobile and wireless devices, game consoles and accessories
- WOW is eXpressDSP™ compliant on both the TI 54x and 55x DSP platforms



"SRS Labs' WOW technology is an exciting addition to our platform and will enable our customers to increase the quality of their audio applications."
—Dennis Barrett, Texas Instruments

Contact us to see how WOW™ can improve your product's audio
Call 1-800-243-2733 or email sales@srslabs.com



©2001 SRS Labs, Inc. All rights reserved. SRS, WOW and the SRS symbol are trademarks of SRS Labs, Inc.



meow?



A software-based statistical real-time profiler can help you vastly improve the performance of TMS320C62x DSP code.

Real-Time Profiler Aids Code Optimization

By Konstantin Merkher
and Jacob Bridger

Optimizing software code to boost the performance of an application is one of the greatest challenges in writing real-time DSP software. At Surf Communication Solutions, we've found that the impact of effective code optimization can be dramatic: We've achieved remarkable performance gains of several orders of magnitude during a project's development cycle.

DSP software development follows Pareto's rule, also known as the 80/20 rule. Translated into software terms, it stipulates that 80 percent of the DSP resources are used by less than 20 percent of the code. For our highly complex signal-processing applications, the rule is closer to 95/5. That phenomenon is extremely encouraging: It means that isolating and optimizing the voracious 5 percent of code reduces DSP MIPS usage and boosts application performance.

You can develop finely tuned, tight DSP code by identifying the few sections that overextend the MIPS budget. One tool commonly used for that purpose is a hardware profiler that is part of or added to an in-circuit emulator. A hardware solution, however, has two drawbacks: expense and

Listing 1: Example ISR code to service timer interrupt for C62x

```
extern cregister volatile unsigned int IRP;
    /* Interrupt Return Pointer */

volatile unsigned short int profiler_enable_flag=0;
unsigned int profiler_program_counters[MAX_DATA_BUFFER_SIZE];
    /* Record Buffer */
unsigned short int profiler_program_counters_index=0;
    /* record Index */

void interrupt c_intXX()
{
    if (profiler_enable_flag)
    {
        if (profiler_program_counters_index<MAX_DATA_BUFFER_SIZE)
        {
            profiler_program_counters[profiler_program_counters_index++]=IRP;
        }
    }
}
```

obtrusion. Buying the tool for one development station may not be a major expense for a typical company, but purchasing it for dozens is typically too great an expense. Moreover, hardware-based solutions may add extra noise and slow down full-speed, real-time processes, skewing the true profile and possibly preventing the application from running correctly in real time.

To overcome those drawbacks,

we've developed a software-based statistical real-time profiler for internal use. It's easy to implement and requires no additional hardware.

THE CONCEPT

The fundamental concept of the statistical real-time software profiler is to take periodic snapshots of the DSP's instruction pointer (program counter). The captured information

shows where the DSP spends most of its MIPS resources. Over time, these periodic (but random) snapshots typically converge on the true distribution function of the application. To learn how much processing time a DSP spends in each software function, you need to write an interrupt service routine (ISR) to handle the timer interrupt and sample the instruction pointer.

You can program the internal timer on the Texas Instruments TMS320C62x generation of DSPs to provide interrupts at given intervals. The actual rate of interrupts depends on the resolution and the speed of profile convergence you want. The ISR must find the return vector and record or store it for future use (Listing 1). However, the interrupts should occur infrequently; otherwise, they'll hamper the smooth operation of the application. The key to obtaining an accurate profile is to run the application for a long time.

Although software development tool sets with profiling options already exist, almost none operate in real-time systems. Such profilers consume large amounts of DSP resources, and the local system can't synchronize with the host system because it can't manage the real-time interface under the added burden. In contrast, DSP overconsumption is insignificant in our system because it adds only 0.1 percent overhead to the main application.

Software-based profilers do have limitations, though. For one, if you program the timer to perform interrupts at short intervals and the time slot is too short, the profiler disrupts normal operation of the application. The longer the interrupt interval, the more time the system needs to get an accurate picture. We find that a 0.1-ms frequency is sufficient for modem, voice-over-IP (VoIP), and fax-over-IP (FoIP) applications that have computing cycles

Listing 2: Example map file symbol list

```
007e1b58 _Error_Control
006abd3c _ExternalBufTable
007d6ba4 _ExternalChannelAddress
006ec530 _ExtractNum
0070ac78 _ExtractRTPHeader
007daff4 _ExtructChannel
007d3f68 _ExtructChannelFromLargeBlockList
007d3438 _ExtructChannelFromList
007d3490 _ExtructChannelFromSmallBlockList
0076db58 _FAX_ANS_Get_Size_Of_Obj
0076daf4 _FAX_ANS_get_state_variables
0076d93c _FAX_ANS_init
0076aa00 _FAX_ANS_modem_main
006d8c44 _FAX_DP_version
0075b9f8 _FAX_GetSizeOfObject
0076d930 _FAX_ORG_Get_Size_Of_Obj
0076d900 _FAX_ORG_init
0076a8e0 _FAX_ORG_modem_main
```

ranging from 5 to 30 ms.

What's more, since our statistical profiler converges over time, it isn't appropriate for tracking MIPS demands that occur infrequently or during initialization. In that case, you can write special support routines that loop the routines many times until the profiler converges.

IMPLEMENTING THE PROFILER

The C62x DSP is equipped with programmable timers. Some of the timers are active and perform various functions while the application is running; others are dormant and available for use. To implement a software-based profiler, the DSP must have an unused timer that is capable of implementing interrupts.

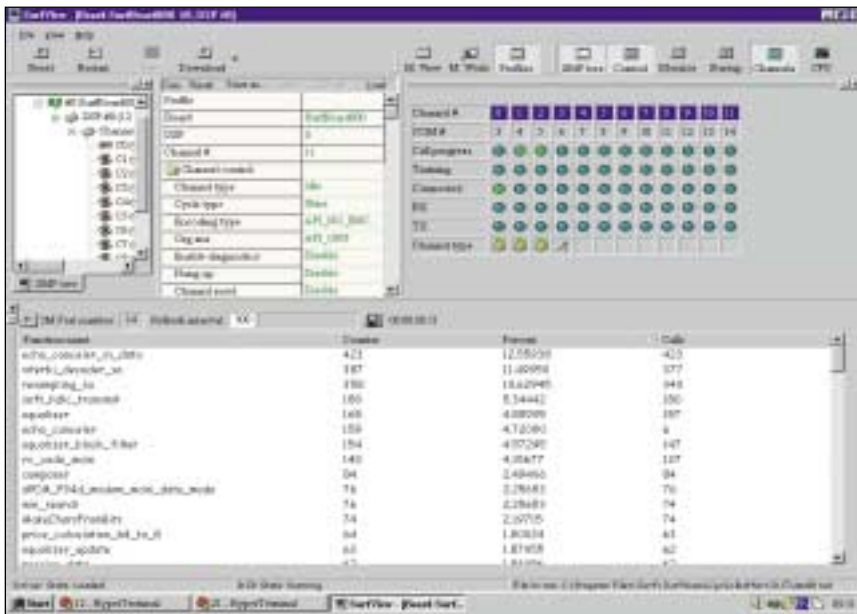
In addition, the host system should be able to collect the recorded entries from the DSP and decode memory addresses (according to the function names to which they belong). To do so, the host system must initially parse a map file produced by a linker. When the system is running, the DSP sends the

recorded addresses as packets. The addresses point to the functions, and therefore their usage value (in percentage points) can be updated or recalculated. The output is a table that includes the names of functions and a percentage value of CPU consumption.

The process can be implemented on any host platform. We use Windows NT as a development platform for the profiler host and internally developed C62x-based PCI target boards for the target software.

IMPLEMENTING THE DSP SIDE

The ISR identifies the returned vector and records it in a buffer for future transfer to the host. A special flag in the host or target API indicates whether recording is enabled or disabled. By using the flag, the host application can isolate specific function branches of the overall target application to be profiled. Although enabling and disabling interrupts could also achieve that, the method isn't recommended, since it synchronizes code sections with the timer.



The profiler's recalculated values should be displayed in a table of at least two columns. The first column lists the function names, and the second the percentage value of DSP MIPS consumption. Arranging the table using the values in column 2 in descending order shows the functions that the DSP spends the most time on displayed first.

The law of averages permits convergence only if the timer and application remain uncorrelated.

Every CPU can receive interrupts and store the return vector in a specific manner; some have a special-purpose register—in the C62x device, the interrupt return pointer (IRP). The addresses can be accessed and recorded. To prevent the host system from being overburdened with profile data, you should use a buffer for recording addresses and transfer them to the host in extended time cycles. Our addresses, for example, are transferred to the host every 10 ms.

When the C62x DSP target software is compiled and linked, several files are produced, one of which is a map file. The map file contains the entire target memory map, including symbol information (such as global variables and function names) and the addresses of the

symbols relative to the physical memory map (Listing 2). The symbols are arranged in ascending order, so that the last address of a function is the one before the first address of the next function. Once a map file is produced, initial parsing can be performed.

After recording the vectors and receiving the buffer of addresses, the host begins the decoding process. It selects each address from the buffer and finds the specific function it belongs to (relative to the memory map). It then increases the access counter of that specific function, enabling the profiler to recalculate the relative DSP MIPS consumption value of the function. The value is calculated by dividing the number of hits for a specific function by the number of received addresses.

The host system periodically receives packets of recorded addresses through the DSP or host

interface. Therefore it must be able to receive blocks of data according to a preconfigured size.

After the decoding phase and matching the instruction counter data with the map file addresses, the recalculated values are displayed in a results table with two or (optionally) more columns. The first column displays the function name and the second displays the percentage value of DSP MIPS consumption (see the figure). The data should be arranged in descending order, according to the percentage of the accumulated run time. Thus the function that the DSP spends the most time on is displayed first.

Our mature development environment—including a highly functional and robust Windows NT-based host control and monitoring application—greatly facilitated implementing the profiler. Using it, our R&D team optimized the most widely used functions in the Surf Multi-access Pool (SMP) application for terminating the V.90 modem, G.7xx VoIP, and V.17/T.38 FoIP. As a result, more channels were able to run simultaneously on a single DSP chip. For example, in the case of Viterbi decoders consumption decreased by one half in the modem data pump, enabling us to reach our target of 15 fully convergent channels on a single TMS320C6202 DSP.

Konstantin Merhker (kostikm@surf-com.com) is a software engineer for Surf Communication Solutions, Ltd. in Yokne'am, Israel, responsible for system analysis and the optimization of Surf's products and solutions.

Jacob Bridger (jbridger@surf-com.com) is Surf's vice president of corporate marketing. He has 15 years' experience in R&D and management of DSP-based real-time embedded software and signal-processing projects, spending the last several years in global high-tech business development.

Continued from page 6

DSP Helps Kodak With Upgradable Product

A TMS320DSC21 DSP from Texas Instruments, Inc. (Dallas, Texas; www.ti.com) sits at the heart of Kodak's mc3, a multifunction consumer imaging and audio product that captures video, still images, and audio. The chip lets Kodak customers upgrade the product, via software downloads from the Web, with the latest audio and video compression formats. The mc3 can record video at 20 frames/s for the highest resolution or 10 frames/s for virtually unlimited recording to removable memory cards. It also captures still color images having VGA resolution and can store up to 90 minutes of MP3 music on a 64-MB CompactFlash memory card.

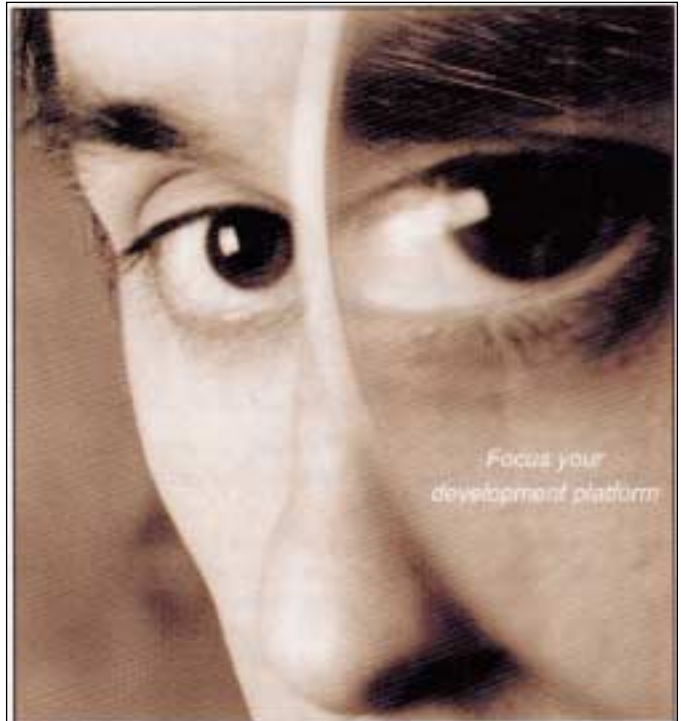


TI Launches On-line DSP Newsletter

Texas Instruments, Inc. (Dallas, Texas; www.ti.com) has started an on-line monthly newsletter called e-Tech Innovations, Digital Signal Processing Edition. Readers can subscribe at www.ti.com/sc/docs/dsps/etechdsp.htm for an easy way to keep informed of the latest DSP news and trends from TI.

PCTEL and Groupe SAGEM Collaborate

PCTEL, Inc. (Milpitas, Calif.; www.pctel.com) has formed a strategic alliance with Groupe SAGEM (Paris, www.sagem.com), France's second-largest telecommunications equipment maker. The aim of the deal is to develop a reference design for use in digital TV set-top boxes. The design will feature PCTEL's Solsis embedded modem for accessing the Internet and include the TMS320C5000 DSP platform. The set-top boxes will be sold or licensed to European television service providers, like France's Canal+, which will then sell or rent them to customers.



"Get to market

Faster

with a DSP focused LINUX solution™

DSPLinux™ lets embedded engineers quickly develop applications even before the hardware is available. It's the Linux solution that leverages the power of Texas Instruments' DSP+ ARM architectures, delivering the performance-leading platform for wireless, multi-media and broadband appliances. Find out more about the DSPLinux SDK and its Appliance Simulator by visiting www.ridgefun.com or calling 208.331.2226 today.



Boise • San Jose • Austin • Osaka • Dublin

© 2001 RidgeRun, Inc. All rights reserved. RidgeRun and DSPLinux are trademarks of RidgeRun, Inc. Linux is the registered trademark of Linus Torvalds in many countries. It is used by RidgeRun under license. All other products and trademarks mentioned herein are the property of their respective owners.

Experts Answer Your Questions

Using Code Composer, can I debug a target board containing two DSPs of different platforms in a single JTAG scan path?

★ In this case, you'll need to launch two separate instances of Code Composer to support each of the DSP platforms. Two separate directories should be created for Code Composer files; the set-up utility will need to be run in each of these directories, and the DSP not being targeted in one instance of Code Composer should be bypassed. Do the same for the remaining DSP. Bypassing DSPs and scan chain devices is discussed in Chapter 1, "Setting Up Code Composer," of *Code Composer User's Guide*.



Should I use the interrupt keyword when implementing an interrupt service routine in a DSP/BIOS application?

★ You can't use the C compiler's interrupt keyword in DSP/BIOS programs. DSP/BIOS interrupt routines must be written in assembly language and must use the `HWI_enter` and `HWI_exit` macros. The C6000 version of DSP/BIOS has an interrupt dispatcher that allows you to write interrupt routines in C. You can also write a C interrupt service routine by making a small `.asm` file that includes just `HWI_enter`, `call cfxn`, and `HWI_exit`.

Can I define my own linker command (.cmd) file instead of one created by the DSP/BIOS configuration tool?

★ Since the Code Composer Studio build tool allows only a single linker command file per project, the best approach is to list the DSP/BIOS linker command file at the top of the user-defined linker command file. To list the DSP/BIOS linker command file in the user defined CMD, add the following line to the top of the file (replacing it with the actual design name):

```
-l yourappcfg.cmd
```

Can DSP/BIOS run on the simulator?

★ Yes, DSP/BIOS runs on the simulator. The simulators currently do not contain a timer interrupt source, so the clock (CLK) and the periodic function (PRD) are effectively disabled.

What is the relationship between CIO's malloc/free and MEM_alloc and MEM_free?

★ DSP/BIOS overrides the standard `malloc` and `free` functions with calls to `MEM_alloc` and `MEM_free`. The segment allocated by `malloc` is controlled by the segment for `malloc()/free()` inside the MEM Manager properties.

How much memory does the memory management system require?

★ As long as no heaps are defined, no memory is used by the MEM Module. If your application requires dynamic memory allocation, a small number of words are required for each heap defined. Beyond that, only memory defined as a heap is required.

How can I control in what memory sections DSP/BIOS objects are placed?

★ The DSP/BIOS Configuration tool lets you place all the objects in different memory locations declared in the Memory Manager through each manger module.

Can DSP/BIOS run in extended memory on C54x processors?

★ Yes, the DSP/BIOS Configuration tool allows you to select the appropriate library under Global Setting. DSP/BIOS requires that the `bios`, `.sysinit`, and `.vect` sections be placed on the overlay (OVLY=1) section of memory (0x000000[EN]0x008000). These sections contain wrappers to support extended memory and are expected in the start-up sequence. All other sections and objects can be placed anywhere in memory. For more information on extended memory with DSP/BIOS, go to www.ti.com/sc/docs/apps/dsp/tms320c5000app.html and locate document SPRA599.

High-Density G.726 Vocoder

Available for TMS320C54x and C6000 DSPs, G.726 vocoder software compresses 64-kb/s packet voice data for 40-, 32-, 24-, or 16-kb/s rates. It can implement 20 channels on a C5400 using 5 MIPS and up to 190 channels on a 300-MHz C6203. The vocoder is available in versions that comply with the TMS320 DSP Algorithm Standard or MSP Consortium M.100, as well as on the company's MSP Media Gateway DSP boards. Licensing fees are \$20,800 for the object code and \$26,000 for limited-use source code. **Commetrex Corporation**, Norcross, Ga.; (770) 449-7775, www.commetrex.com

eXpressDSP-Based Library

Version 5 of SigLib, a highly portable ANSI-C source DSP library, touts compatibility with the TMS320 DSP Algorithm Standard. It includes many of the low-level routines used in today's telecommunications algorithms and accommodates many of the fundamental telecom operations found in modems, mobile phones, and other network access devices. It comes with comprehensive examples and documentation and sells for \$350. **Numerix, Ltd.**, Leicestershire, U.K.; +44 (0)-7050-803996, www.numerix-dsp.com

DSP Development Kit

The Developer's Kit for Texas Instruments Digital Signal Processing combines MATLAB 6 and Simulink 4 with eXpressDSP Real-Time Software Technology to simulate, generate, and validate designs build around TMS320C6000 and C5000 DSPs. Features include MATLAB links for Code Composer Studio

and Real-Time Data Exchange and Simulink targets for CCS and the TMS320C6701 EVM. The kit is available for Windows 95, 98, and NT and works with CCS version 1.2. Prices start at \$1,000 for an individual PC license. **The MathWorks, Inc.**, Natick, Mass.; (508) 647-7589, www.mathworks.com

DSP Imaging Evaluation Kit

A tool for building real-time audio and video compression applications, the Imaging Evaluation Kit addresses four phases of development: evaluating available technologies, assessing a DSP platform's suitability for an application, functional prototyping, and bringing re-designed systems to market quickly. A basic version sells for \$2,995 and includes a TMS320C6111-based board and drivers, sample algorithms, and Code Composer Studio. Another version, which adds a camera, microphone, and speakers, sells for \$6,495. **A.T.E.M.E.**, Velizy, France; +33-1-46-01-55-72, www.ateme.com

USB Emulator for TMS320C54x

The SB-USB, a self-powered high-speed emulator, connects to a PC's USB port to debug systems built around one or more TMS320C54x DSPs. Featuring two programmable counter-timers, it occupies a 4- x 2.5-in. circuit board and operates seamlessly with Code Composer Studio. The emulator sells for \$3,000 and includes cables, software drivers, and a user manual. Custom versions are available. **Domain Technologies**, Plano, Texas; (972) 578-1121,



NFO Evaluation Board

An evaluation board that operates with a three-phase induction motor takes advantage of the natural-field-oriented (NFO) control algorithm stored in the flash memory of TMS320LF2406A 40-MIPS DSPs. The algorithm gives accurate, sensorless torque control over a wide speed range. The board can work in a stand-alone mode, with speed or position sensors feeding on-board control loops, or connected to a PC through an optically coupled serial link. Available in July and bundled with a Labview user interface that controls the motor and modifies motor and control parameters, it sells for \$500. **NFO Control AB**, Lund, Sweden; +46-46-286-29-26, www.nfo.se

Prototyping Daughterboards

Two prototyping daughterboards, ProtoPlus and ProtoPlus Lite let developers construct a prototype circuit that plugs into the TMS320C2000 and C6000 DSP platforms, the C54x, and the DSK and EVM development systems. The boards give access to all signals and power rails. They accept external ± 12 -V power. The ProtoPlus Lite, a two-layer board, sells for \$125. The ProtoPlus, a six-layer board, has separate ground, and 3.3- and 5-V planes; it sells for \$225. **DSP Global, Inc.**, Warwick, R.I.; (401) 737-9900, www.dspglobal.com



Single-Board Media and Signaling Gateway

SuperSpan II is a completely integrated single-board embedded hardware and software platform for implementing carrier-class media gateway, SS7 signaling gateway, cellular infrastructure, and unified messaging systems. Included are octal T1/E1 network access ports, i860 signaling controller, PowerPC 750 protocol controller, high-density DSP resource mezzanine board, and embedded software for convergent voice and data applications. Besides sporting VoIP, wireless, V.90, and G3 fax software for the TMS320C5000 DSP platform, SuperSpan II embeds H.323, MGCP, SIP, TCP/IP, SS7, and



TCAP/ISUP signaling stacks and internetworking functions. Prices start at \$3,000. **Voiceboard Corporation**, Oxnard, Calif.; (805) 985-6200, www.voiceboard.com

Assistant Enhances Development Tools

Development Assistant for C works independently or alongside Code Composer Studio. The assistant uses an ActiveX interface to communicate with Code Composer Studio and debugger commands. Among its features are an editor with structured and nonstructured flowcharts, start debugger commands for Code Composer Studio, symbol browser, call- and type-hierarchy graph,

makefile generator, software metrics, interface to version control systems, project manager, and static code analyzer. Starting prices range from \$295 to \$660 each, depending on the target DSP. **RistenCASE GmbH**, Wallisellen, Switzerland; +41-1-883-35-70, www.ristancase.ch

IP Phone Chip

The IP Phone, a chip built around a 100-MHz TMS320C549, delivers the features of a standard PBX-based phone, including call transfer and caller ID. Accompanying software includes G.723 and G.729 voice compression and automatic echo cancellation algorithms, as well as IP, UDP, RTP, and DHCP



New ActiveX Plug-In for Code Composer Studio!

- ✓ Adds Flow Charts, Call- and Type- Hierarchy Graphs and Software Metrics.
- ✓ Increases software development productivity up to 50%.

RistanCASE GmbH
Zielackerstrasse 198304 Wallisellen, Switzerland
<http://www.RistanCASE.ch/da-c>



TORNADO E-SERIES

The Choice For Embedded DSP

Available with:

TMS320C6xxx/5xxx or
TMS320VC33

USB-Dual RS232/422-GPIO

\$739 to \$1,854 each



www.microlabsystems.com

network protocols; SIP signal protocol; and DES encryption. Prices start at \$300 each. **ADtech**, Bonnelles, Belgium; +32-4-338-13-30, www.adtech.com

Low Data-Rate AMBE+2 Vocoder

A low-data-rate AMBE+2 vocoder operates at 2.0 to 9.6 kb/s for applications where low bandwidth and high-quality speech performance are high priorities. Its model-based multiband excitation algorithm carries distinct advantages over conventional CELP-based vocoders, including higher mean opinion scores. The software runs on the TMS320C5000 DSP platform. The price depends on customer speci-

cations. **Digital Voice Systems, Inc.**, Burlington, Mass.; (781) 272-5606, www.dvsinc.com

Ada Suite Teams Up with TI DSPs for Military Service


The 83/95 Ada Compiler, an Ada software suite that targets Texas Instruments' SMJ32C6000 DSP platform for military applications, includes a full Ada symbolic debugger and Ada run-time options that include tight integration with the DSP/BIOS kernel. The suite fits tightly into Code Composer Studio. It works with SMJ32C6000, SMJ32C6201, and SMJ32C6701 DSPs and runs on Windows 95, 98, 2000, and NT machines. The 83/95 Ada Compiler costs \$25,000 for the first

seat and \$20,000 per seat thereafter. A maintenance program costs \$5,000 annually. **Irvine Compiler Corporation**, Irvine, Calif.; (949) 250-1366, www.irvine.com

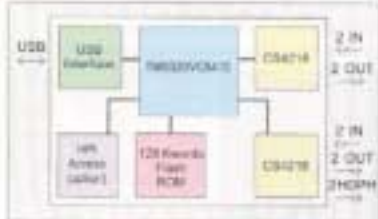
Libraries Tuned to C6000 DSP Platform

The GD-100 DSP vector library for the TMS320C6000 DSP platform comprises over 100 functions and macros, including transforms, filters, and vector operations. The GD-200 math library for the C67x consists of algebraic and trigonometric functions and utilities. They sell for \$3,500 and \$2,450 respectively. **Kane Computing**, Cheshire, U.K.; +44 (0) 1606 351006, www.kanecomputing.com



Audio4-5410 DSP Data Acquisition



- 4 Input + 4 Output audio channels
- 16 bit AD/DA, 8 kHz to 48 kHz High-speed USB interface
- Real-time Code Composer Studio™ support
- 100 MHz TMS320VC5410 with 64 Kwords Internal RAM
- VAB5000 compatible



TI Third Party Network Member

1-888-828-3543
www.domaintec.com

Q. Who offers the industries most comprehensive suite of embedded communications software?

A. GAO Research Inc.

INTERNET APPLIANCE SOFTWARE

Broadband Modems	Fax & Fax Relay	Telephony
ADSL G.Lite	V.34 - V.17 V.29 V.21 Ch.2 V.27ter	Line Echo Canceler Acoustic Echo Cancel Automatic Gain Contr
Voice band Modems	T.4 T.30 T.37 T.38	Call Progress DTMF Caller ID Type I & II Voice Activity Detect Comfort Noise Generat
V.92 V.90 V.34 V.32bis V.32 V.22 V.22bis V.23 V.21 Bell 212A Bell 103	Speech G.729A G.729B G.729 G.726 G.711 G.723.1 G.722	

NETWORK INFRASTRUCTURE SOFTWARE

Residential Gateway	VoIP	RAE
ADSL Voice over DSL Fax over DSL fallback to V.92	LEC, VAD, CNG, Vocoder Time-Detection/Generation, Loss packet detection & reconstruction Jitter buffer & sequencing	Package for remote access servers include multi-channel V.92 VoIP & Fax over IP

DSPs, MICROPROCESSORS, RTOS & APPLICATIONS SUPPORTED

Texas Instruments - TMS320C5000 & TMS320C6000 DSP Platforms
Integrates with MP3, JPEG, MPEG, TCP/IP, most popular RTOS and web browser



GAO Research Inc.
info@gaoresearch.com
www.gaoresearch.com
416-292-0038
#1 In embedded Communications Software



Needed: New Compilation Tools to Help Optimize Embedded Code

By Alan S. Ward

For embedded software programmers, the performance of code generated from a high-level language is becoming increasingly important. At the same time, code size is often a critical concern. Three trends—the leaps in the capability of embedded processors, the significantly greater complexity of embedded software, and the mushrooming of a variety of handheld devices—are driving a huge and rapidly expanding need for tools to help automate programming. In order to keep up with these changes, embedded software programmers clearly need a set of accessible and easy-to-use tools to optimize their code for performance and size demands.

Many embedded compilation tools already possess the sophistication to highly optimize applications. The difficulty is extracting that capability from the tools. Consider the task of scheduling code for a VLIW DSP that supports eight parallel instructions per cycle. Applying “parallelism-generating” transformations results not only in faster code, but also in a larger code size. In the embedded environment, though, programmers are usually limited by real-time and cost constraints—that is, by cycle counts and code size.

Because of these conflicting con-

straints, most compilers for embedded processors contain some mechanism for controlling the optimizations that affect size versus speed. However, simply managing this mechanism can be a daunting challenge. Given the simplified situation of a compiler switch with two states—best performance or best code size—and a very small application with 20 units of code (such as a file or function), 2^{20} —or roughly 1 million—combinations would exist.

A more realistic situation would involve a compiler switch with many states along a size-to-speed continuum and hundreds of code units. Obviously, programmers can't search the entire solution space for the optimum mapping of options to code units. Instead, they typically use the 80/20 rule, which states that 80 percent of an application's cycle requirements are in 20 percent of the code. The exact percentages can be debated, but the premise is usually true. Using this rule and knowledge of the application, programmers can quickly prune much of the solution space. Indeed, such a manual, iterative process is still used to determine a satisfactory solution. What's more, this problem is only one example; developing production-quality embedded code with

old compilation tools involves many trial-and-error processes.

Fortunately, compiler tools are appearing that address this level of interaction between the tools and their users. A profile-based compiler uses criteria supplied by the programmer to automatically build and profile multiple sets of options for coding all the software's key functions; then it plots the most favorable option combinations on a curve that represents different trade-off values between performance and code size. Using the graph, programmers can select the right trade-off for the design's requirements, like the tightest code for a given cycle count or the fastest execution at a given memory size. Thus this type of compiler can save weeks of effort and assure developers that they have the best solution for reconciling performance with code size.

Other examples of new or upcoming compilation tools include ones that structure code sequences to better map to the underlying processor and ones that experiment with the memory layout of code or data to utilize on-chip memory or cache most effectively.



Alan Ward is the C6000 DSP compiler tools manager and a distinguished member of the technical staff at Texas Instruments, Inc. in Houston.

xICE* Multi-Target Emulator

DSP/RISC Debugging

In the past emulators provided a physical and software link between a *single* scan chain target DSP/RISC and a host computer, running a Debugging Interface Program.

Until now, multi-processor systems have required that devices be daisy chained into the one scan path. The longer scan path does not allow full scan rates because of scan path delays through the respective processors (switching in and out of bypass) and PCB-processor interconnects.

xICE* Solution

The xICE* Multi-Target Emulator is capable of emulating multiple DSPs/RISCs on separate scan paths **simultaneously**. This is particularly useful for parallel processing applications employing multiple DSPs and/or RISCs, such as image processing, radar/sonar systems, and processor arrays.

The xICE* consists of one PCI host card and multiple pods, one for each of the separate scan paths. This enables a number of DSPs / RISCs, each on separate scan paths, to be debugged from one PCI slot, all at full scan rates upto 30MHz, all at the same time.

Softronics Emulators

- xICE* -> PCI Bus Multi-Target Emulator
- nICE -> 100baseT Ethernet Emulator
- DspIcE* -> PCI Bus Emulator
- Ice*Pack -> ISA (16 bit) Bus Emulator
- Mice*Pack -> Parallel Port Emulator



Softronics

inform@softronx.com
www.softronx.com

Ph+61 500 505059
Fx+61 500 505049

Processor Support

- TMS320C2xxx <-> TMS320C3x <-> TMS320C4x <-> TMS320C5x
- TMS320C54xx <-> TMS320C55xx
- TMS320C62xx <-> TMS320C67xx <-> TMS320C64xx
- ARM7 <-> ARM9 <-> TMS470 <-> TMS320CAVxxx
- Flash Programming Support <-> RTDX <-> DSPBios

Texas Instruments trademarks include RTDX(TM) DSP/BIOS(TM) Kernel, Code Composer Studio(TM) IDE and these DSP generations: TMS320C62x(TM), TMS320C67x(TM), TMS320C64x(TM), TMS320C54x(TM), TMS320C55x(TM), TMS320C2800(TM), TMS320C5x(TM)

Software Support

- TI Code Composer Studio
- ARM Aspx
- Eonic Virtuoso

x I C E *

Multi-target DSP/RISC



The thumb is not technically a finger.

Integrated development tools make DSP system programming simpler.

Standards for application interoperability make integrating DSP programs simpler.

Products from a third-party software network make DSP system programming simpler.

A scalable real-time kernel makes DSP programming simpler.

eXpressDSP Real-Time Software Technology from TI.

Simplicity, simplicity, simplicity, and simplicity, right at your fingertips.



With eXpressDSP™ Real-Time Software Technology from Texas Instruments, DSP programming can become simpler than ever before. Code Composer Studio™ integrated development tools give you powerful data visualization, real-time analysis, and code-optimization capabilities. The DSP/BIOS™ kernel provides a scalable, stable real-time foundation upon which you can build all of your DSP designs. Application-interoperability standards help ensure that your software components can be continually reused, without the hassle of redesign. And thanks to our global third-party software network, you have access to a library of modular, interoperable, off-the-shelf algorithms and plug-ins that let you spend more time designing and less time reprogramming. To put the power of simplified DSP programming at your fingertips, rely on eXpressDSP Real-Time Software Technology, only from Texas Instruments.

For an eXpressDSP developers info pack that includes free Code Composer Studio evaluation tools and information on eXpressDSP-compliant third-party products, go to www.dspvillage.ti.com/simplicity

eXpressDSP, Code Composer Studio, DSP/BIOS, and the red/Black logos are trademarks of Texas Instruments. 21-4083 © 2006 TI

THE WORLD LEADER IN DSP AND ANALOG



TEXAS
INSTRUMENTS