

EFFICIENT MULTIRATE SIGNAL PROCESSING WITH THE C55X

With its dual MAC, abundance of buses, and copious internal RAM, the TMS320C55x allows a high degree of parallelism—just the ticket for applications using decimating filter banks.

By Michael Tsirounnikov

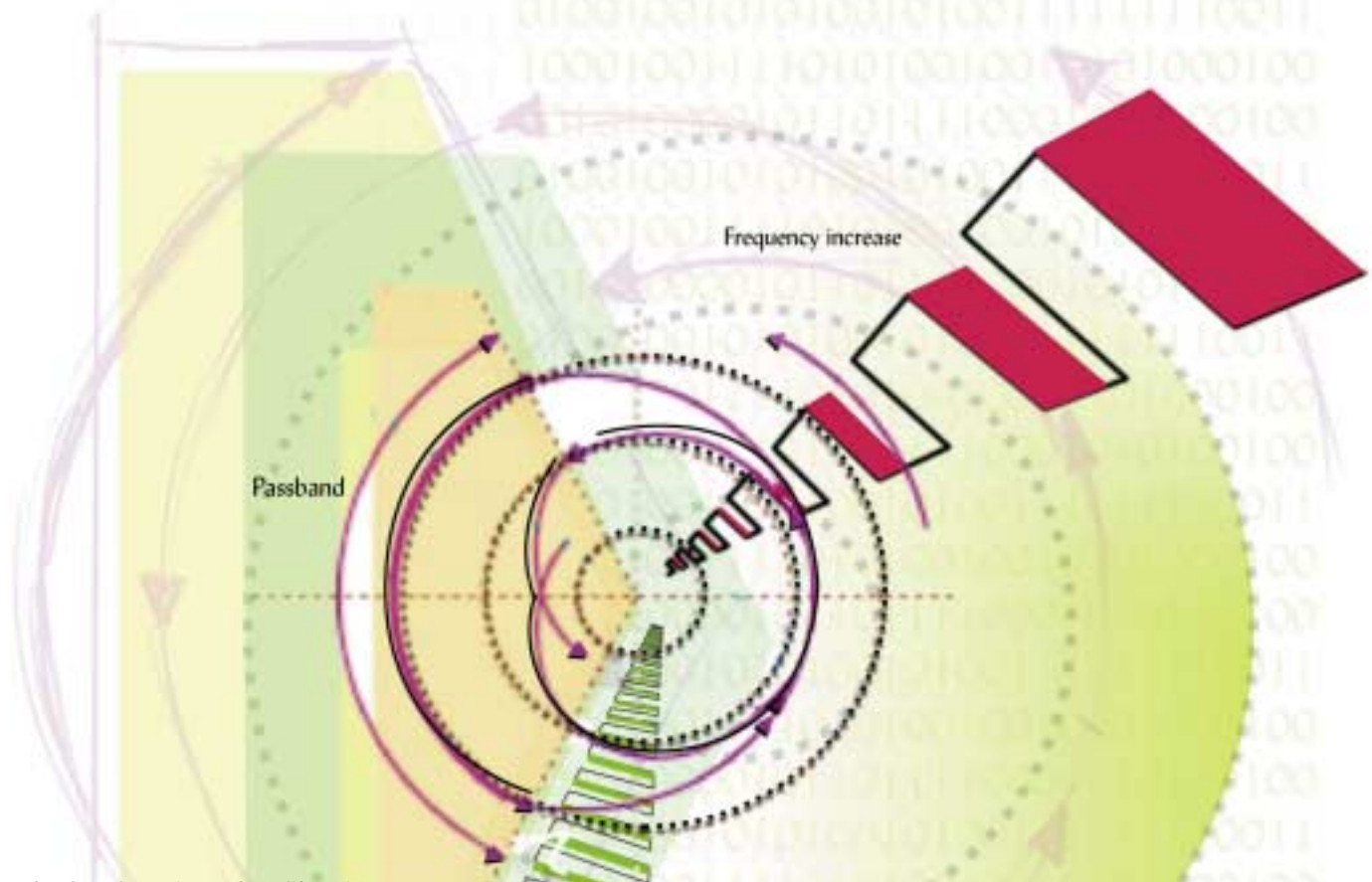
Listing 1: Definition of the Structures

```

; constants
APP_FB_FSZ      .set 100      ; application-specific filter size L
APP_FB_FLS      .set 6        ; application-specific filter bank size N
;----- structures
APP_tOut        .struct
; something ...
sS              .int          ; os(i)
sA              .int          ; oa(i)
s1Energy        .long
; etc, etc, etc
APP_tOut_Sz     .endstruct
;
APP_tSc         .struct      ; scratchpad
; something...
aOut            .int (APP_tOut_Sz*APP_FB_FLS)
asBpData        .int APP_FB_FSZ      ; x(t)
asPlus          .int APP_FB_FSZ/2    ; c(t)
asMinus         .int APP_FB_FSZ/2    ; s(t)
; etc, etc, etc
APP_tSc_Sz      .endstruct

```

Compared with previous generations of digital signal processors, Texas Instruments' TMS320C55x generation of DSPs is particularly well suited to multirate operations in the complex domain. Its single-cycle dual multiplier/accumulator (MAC), multiplicity of buses, and generous endowment of internal RAM give it about an order-of-magnitude cumulative improvement in performance over previous generations of DSPs when implementing signal detectors, decimating filter banks,



and other functions that filter input data through an array of same-length complex filters.

Complex-domain operations make it possible to overcome the limitations of real-domain bandpass sampling, thereby helping to maximize a filter's decimation ratio without any loss in precision. A decimating filter bank can be viewed as an application-specific transform or a signal compression technique in which blocks of real signal data are converted into shorter sets of down-sampled values.

PROCESSOR HIGHLIGHTS

Compared with its predecessors, the TMS320C54x generation, the C55x is highly advanced. Most notably, just its single-cycle dual MAC, supported by multiple buses that let it retrieve three 16-bit data words in a single cycle, gives the processor a twofold performance advantage. In addition, the new

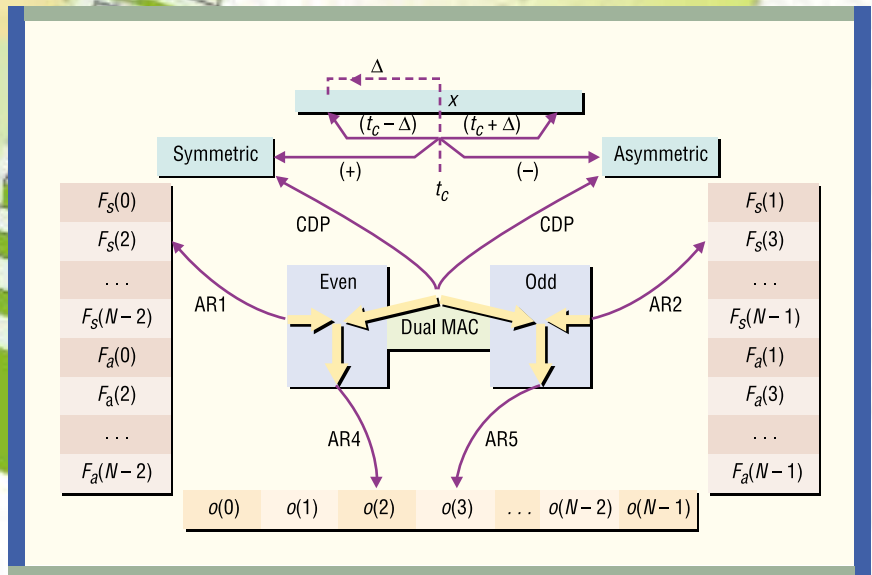


Figure 1. Organizing the data into blocks of symmetric and asymmetric coefficients and then taking them in sequential order for the even and odd indices, respectively, keeps auxiliary pointer manipulations to a minimum. The coefficient data pointer (CDP) points to the prefolded input data; the auxiliary registers (ARx) point to the coefficient arrays.

processors have significantly more internal RAM. These resources can be used very effectively to realize various filter banks and functional transforms.

Relative to code written for the C54x, code for the C55x has less loop initialization and finalization overhead—an important advantage in staged multirate signal processing, when filters are often short and the overhead can therefore add significantly to the execution time.

Other valuable new features of the C55x are its coefficient data pointer (CDP) register and its auxiliary registers (ARx), which now allow long immediate offset values to be included in the op code without adding any extra cycles when the program is executing. This capability makes it easier to develop assembly code in a C-like fashion, thereby shortening the R&D cycle.

The key to the successful design of a DSP application is appropriate data organization for both the filter bank coefficients and the input data. Properly organized data makes for a smooth, natural flow of operations with minimal pointer manipulations and no unnecessary discontinuities in the program flow.

To exploit the advantages of complex-domain filtering, the input data vectors should be decomposed into symmetric and asymmetric components; the decomposition helps to boost performance as much as twofold. The new CDP register is exploited in this connection, being used to point to the decomposed data, while the auxiliary registers point to the filter banks.

An obvious drawback of multirate signal processing is its need for a large amount of fast RAM in which to store filter bank coefficients. That need usually won't be a problem for the C55x generation because most of the processors have more than enough internal RAM. Also, their speed can compensate for the increase in program code by allowing more instances of an algorithm to be run in a given time.

REALIZING A GENERIC COMPLEX FILTER BANK

Transforming real-domain input data into the complex domain simplifies later signal processing, especially if the application makes use of various nonparametric spectrum analysis techniques. Working in the complex domain also helps boost performance. The major MIPS savings obviously come from the decimation process (the filters are run less often), and those savings are directly proportional to the decimation ratio.

The savings arises from the fact that real-domain bandpass subsampling is subject to many constraints [1] that must be strictly adhered to if an application is to operate properly. The key problem is aliasing: for a

Listing 2: Assembly Code for the Filter Bank Function

```
.mmregs
.cpl_on
.arms_off
.text

;-----
.global _APP_filter_bank
_APP_filter_bank
;-----
;---- function arguments:
; ar0 = APP_tSc *pSc
; ar1 = ptr to _aaEvenFilterBlock[0][0]
; ar2 = ptr to _aaOddFilterBlock[0][0]
;

    pshboth(xar5);
    pshboth(xar6);
    bit(ST2, #ST2_ARMS) = #0;
    bit(ST1, #ST1_FRCT) = #1;
; sum / sub -----
    xar3 = mar(*ar0(#(APP_tSc.asBpData+APP_FB_FSZ-1)));
    xar4 = mar(*ar0(#(APP_tSc.asPlus)));
    xar5 = mar(*ar0(#(APP_tSc.asMinus)));
    xar6 = mar(*ar0(#(APP_tSc.asBpData)));
    brc0 = #(APP_FB_FSZ/2-1);
    localrepeat {
        t0 = *ar3-;
        HI(ac0) = *ar6+ + t0, LO(ac0) = *ar6+ - t0;
        *ar5+ = LO(ac0), *ar4+ = HI(ac0);
    }
; filter -----
    xcdp = mar(*ar0(#(APP_tSc.asPlus)));
    xar3 = mar(*ar0(#(APP_tSc.aOut+APP_tOut.sS)));
    t0 = #(-APP_FB_FSZ/2+1);
    t1 = #(APP_tOut_Sz);
    brc0 = #(2-1);
    brc1 = #(APP_FB_FLS/2-1);
    localrepeat {
        localrepeat {
            ac0 = *ar1+ *coef(*cdp+),
            ac1 = *ar2+ *coef(*cdp+);
            || repeat #(APP_FB_FSZ/2-3)
                ac0 = ac0 + (*ar1+ *
                    coef(*cdp+)),
            ac1 = ac1 + (*ar2+ * coef(*cdp+));
            ac0 = rnd(ac0 + (*ar1+ * coef
                (*cdp+t0))),
            ac1 = rnd(ac1 + (*ar2+ * coef(*cdp+t0)));
            *(ar3+t1) = HI(ac0);
            *(ar3+t1) = HI(ac1);
        }
        xcdp = mar(*ar0(#(APP_tSc.asMinus)));
        xar3 = mar(*ar0(#(APP_tSc.aOut + APP_tOut.sA)));
    }
; exit -----
    bit(ST1, #ST1_FRCT) = #0;
    bit(ST2, #ST2_ARMS) = #1;
    xar6=popboth();
    xar5=popboth();
    return;
```

given sampling frequency, f_s , a signal with frequency $f_s/2 + d$ is aliased by a "mirrored" signal, $f_s/2 - d$, for any deviation, d . In effect, the whole frequency plan is folded multiple times within the interval $\{0 \dots f_s/2\}$.

The operations on analytic signals in the complex domain are much simpler, because the frequencies ($k * f_s + d$), for any k , are aliased into themselves—the frequency plan can be viewed as going around in a circle. Consequently, the central frequency of the signal band can be in any relationship to the sampling frequency, and the choice of the decimation ratio is much relaxed, although it's still constrained by the filter properties, the signal/noise spectrum, and the application requirements.

So how, exactly, do you convert a real-domain signal, $X(t)$, into complex-domain signal $o_i(t)$? For:

$$X(t) = \{x(t), x(t-1), x(t-2), \dots, x(t-L+1)\}^T$$

$$o_i(t) = os_i(t) + j * oa_i(t)$$

where L = the length of the filter, what's needed is a complex filter bank made up of N subfilters, Fz_i , each of which has the form:

$$Fz_i = Fs_i + j * Fa_i$$

where F_s indicates real components of Fz , and F_a indicates imaginary ones:

$$Fs_i = \{fs_{i,0}, fs_{i,1}, \dots, fs_{i,L-1}\}^T$$

$$Fa_i = \{fa_{i,0}, fa_{i,1}, \dots, fa_{i,L-1}\}^T$$

$$i = (0 \dots N-1)$$

(all of the subfilters are assumed to be of the same length) and $o_i(t)$ can be given as:

$$o_i(t) = Fz_i^T * X(t)$$

For a very wide class of filter banks, Fz_i can be represented as the sum of two components, a strictly symmetrical real one, Fs_i , and an asymmetrical imaginary one, Fa_i , and both components of each subfilter will still comply with the rules for the Hilbert transform. The straightforward approach to performing these calculations involves using the `firsn()` and `firsn()` instructions of the C55x, but there's a superior approach. If the bank has many subfilters of equal length, it's more efficient to prefold the input data, $X(t)$, around its center point, (t_c) , into two sets, one a sum of the samples $x(t_c - t)$ and $x(t_c + t)$ and the other the difference between those same samples:

$$s(d) = x(t_c + d) + x(t_c - d)$$

$$a(d) = x(t_c + d) - x(t_c - d)$$

With the input data in complex form, our task now is to optimize the processing operations so as to exploit the C55x's dual MAC as fully as possible. Two things that can be done in pursuit of that goal are:

- Use CDP (which is, by default, a pointer to coefficients) as a pointer into the prefolded data arrays, while using auxiliary pointers to operate with coefficients.
- Regroup filter banks. When CDP points to the summed data, $s(t)$, we need both auxiliary pointers to operate with arrays from the Fs_i set. Therefore we can set up two "parallel" blocks of filter coefficients for either Fs_i or Fa_i and use even and odd subfilters sets, $\{Fs_0, Fs_2, Fs_4, \dots\}$ and $\{Fs_1, Fs_3, Fs_5, \dots\}$, linkable into different single-access RAM segments. Then the filtering with Fs_{2i} will be performed simultaneously with the filtering with Fs_{2i+1} . The modifications of operations with Fa_i are similar. Of course, the code

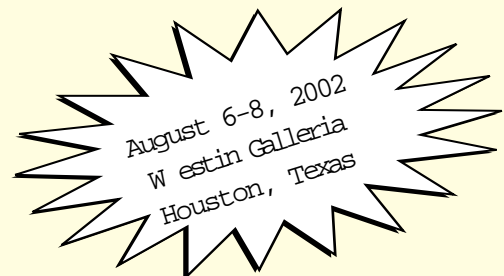


TEXAS INSTRUMENTS

Attend TI's conference for hands-on DSP application training and industry networking.

Perfect for hardware, software, and application engineers, managers, algorithm providers, board developers, consultants and educators.

- Ease product development flow
- Experience hands-on technical training
- Openly exchange ideas, concepts and visionary insights with peers



Sign up today www.tidevcon.com

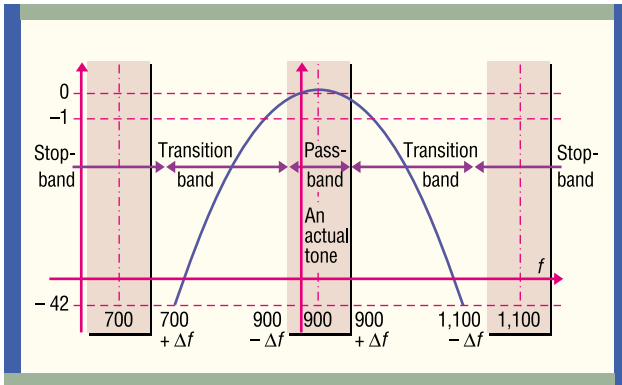


Figure 2. This diagram shows the transmission characteristics of the second-bin (900-Hz) filter before decimation. Both axes are scaled linearly. Ideally, the second filter's stopband attenuation would reach its maximum value over the passbands of all the other frequency bins. The filter's behavior in the transition bands is less important, since the only disturbing factor is circuit noise.

will definitely be simpler if the number of filters in the filter bank is even. This approach becomes more efficient as the number of subfilters grows; indeed, it makes sense for four or more subfilters.

You can use a very similar approach for the more general case of a complex input and a complex filter bank; the processing flow and the code are essentially the same. Be sure to use the Hermitian (conjugate transposed) operator while doing complex domain filtering; otherwise the frequency plan will be inverted.

Determining the best trade-off among decimation filter, block size, and decimation ratio must be decided case by case. Sometimes it makes perfect sense to use longer filters with better shape factors (narrower transient bands), which allow a higher decimation ratio. But since it's impossible to provide general advice appropriate for all circumstances, it's up to you to decide on the trade-offs when you develop your algorithm.

In general, because the filter length is typically longer than the block size, the processing flow involves concatenating incoming data with previously saved block(s) of data and then updating the "saved signal" array. The filter bank function is C-callable:

```
void APP_filter_bank(APP_tSc *pSc, Int *psEven, Int *psOdd);
```

The filter function accepts as parameters a pointer to a scratchpad structure and pointers to the even and odd filter blocks (Listing 1; the code for concatenation

and updating described above has been omitted, since it's straightforward), each grouped into an array of the form:

```
Int_aaAppEvenFilterBlock[2*N/2][L/2];
```

where the first $N/2$ rows of half-sized filters ($L/2$) are taken in sequential order from a symmetric set, F_s , and the rest are taken from asymmetric set F_a , for the even and odd indices, respectively. Organizing the data in that manner (Figure 1) helps keep auxiliary pointer manipulations to a minimum.

DATA FORMAT

In the same manner, the output of the algorithm should represent the data in the format that's most suitable for the further processing. A C-like structure `APP_tOut` is declared, which may contain other fields if required by an application. The function `APP_filter_bank()` only stores $o_i(t)$ (decimated complex signal pairs per subfilter) into predefined locations.

Note that the filter length doesn't have to be even.

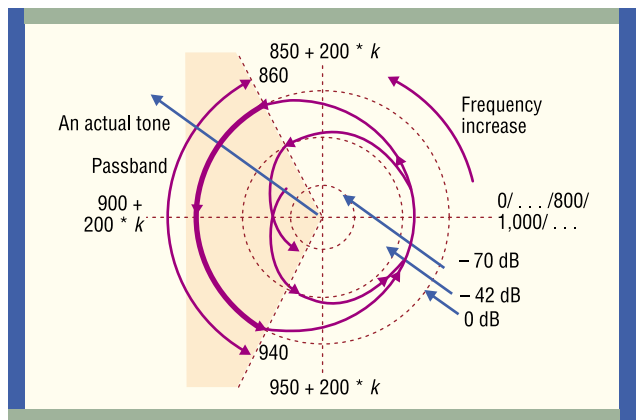


Figure 3. By examining the postdecimation signal and filters using polar coordinates, it's easy to see that the passbands of all the MFR1 bins will overlap after decimation and will fall within the same tinted sector around the vector $(-1, 0)$, which will simplify postprocessing. That won't necessarily be true for other applications.

The code for odd filter lengths is a bit more complicated but very similar.

The assembly code example makes extensive use of the long immediate ARx offset registers, since no extra cycles are required for the C55x and the increase in the code size is acceptable (Listing 2; as with Listing 1, the code for concatenation and updating described earlier has been omitted). The use of these registers

greatly helps with writing, debugging, and testing assembly code by running bit-exact comparisons with a C version of the same algorithm. Despite the relative complexity of the C55x instruction set, I found that I could write C55x assembly code faster than I could write code for previous DSPs, like the C54x or earlier generations.

As you can see from the listing, filtering with F_s and F_a is done using an external loop over the block repeat counter register (BRC0). The pointers to the input array (pSc->asPlus or pSc->asMinus) and the output array (pOut->sC or pOut->sS) are updated when the first iteration is over, but the pointers to the coefficients remain free-running.

Note that the internal loop initiation and finalizing overhead take only three processor cycles. In certain cases it may be possible to decrease the number of cycles even further.

THE FILTER BANK FUNCTION IN ACTION

Let's demonstrate the use of this procedure to detect multifrequency R1 signals as specified in the ITU-T Q.320, Q.322, and Q.323 recommendations. Although a very simple example, it demonstrates the advantages of this approach very nicely.

Valid multifrequency signals are represented by two, and only two, tones from a set of six frequencies $f_i = 700 + i * 200$ Hz, $i = (0 \dots 5)$, with 1.5% tolerance and a duration of at least 60 ms. In this kind of application, noise isn't usually a problem because tone detection is done before a call is established; hence there's no voice on the line to corrupt the tone signal.

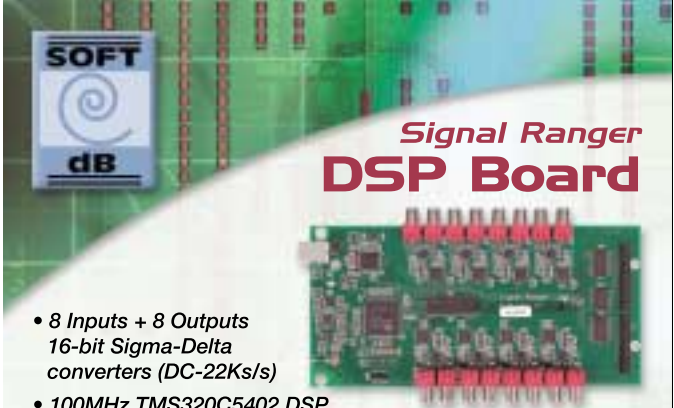
Our filter bank consists of six subfilters of length 100 (input signal sampling frequency $f_s = 8$ kHz), derived from a Hamming prototype and modulated by $\cos(2\pi f_i[t - t_c])$ and $\sin(2\pi f_i[t - t_c])$. (Figure 2 shows the filter shape for the second frequency bin, 900 Hz.) The passband (tinted area) is $\pm\Delta f$ (about 1 dB), $\Delta f = 40$ Hz, and the stopband is $200 - \Delta f = 160$ Hz (42 dB). These values provide some headroom and allow signal processing with a maximum frequency deviation of 2.35% even for the highest-frequency bin, 1,700 Hz.

The choice of the decimation ratio depends greatly on the operations to be performed on the decimated signal. Although signal detection applications allow for deep subsampling, other applications, such as sub-band adaptive filtering, may impose another set of (obviously, much harder) constraints on how low a signal may be decimated. In our application, the transition bands (-160:-40) and (40:160), relative to the central frequencies, may be overlapped (Figure 3).

If we consider the shape of the decimation filter only, the Nyquist barrier can be seen as deliberately broken. A signal with the spectrum falling into transition bands beyond the tinted area will be aliased. Fortunately, no fatal consequences result, because the signal there is only noise and there's no need to reconstruct the original signal. Note that the area of interest (-40:40 off the central frequency, 900 Hz) is "clean." Nevertheless, it's overlapped with the signals in adjacent frequency bins and the accompanying noise, all in the stopband area of the filter. Therefore, to utilize such a high decimation ratio as 40, the prototype filter will be "sufficiently good" and have an adequate shape factor and stopband attenuation.

THE INPUT

The input into the function is a block of 40 samples (appended to the saved preceding data), which is transformed into six complex pairs. This number of samples corresponds to 5 ms, an interval that fits well with other signal processing algorithms, like ITU-T



Signal Ranger DSP Board

- 8 Inputs + 8 Outputs
- 16-bit Sigma-Delta converters (DC-22Ks/s)
- 100MHz TMS320C5402 DSP
- Full-speed USB interface with Independent USB controller
- External bus connectors giving access to Program and I/O spaces of the DSP

Software environment designed to accelerate your DSP development

- USB "Plug and Play" drivers (Windows 95, 98, Me & 2000)
- DLL interface to Visual Studio™ PC applications
- Interface to LabVIEW™ PC applications
- Efficient DSP driver for the analog I/Os (for C or assembly)
- Complete debugger with high-level and graphic features
- Complete user manual including code examples

High-performance/low-cost: 700\$ US

www.softdb.com

TECAL INSTRUMENTS

G.72x vocoders. The further processing of the decimated signal, to estimate the instantaneous frequency and energy in this low-noise case is obviously simple. If you're interested in more advanced techniques, *Modern Spectrum Analysis* [2] provides an excellent starting point.

For this decimation ratio ($dr = 40$), $N = 6$, the filter length $L = 100$, and the sampling frequency $f_s = 8,000$ Hz; the theoretical, straightforward MAC count is $2 * L * N * (f_s / dr) = 0.24M$ MACs. The processing figure for APP_filter_bank is only 0.11 MIPS (550 clock cycles per function call), including the overhead of the function call. That low number clearly demonstrates the superiority of this approach over the straightforward one and, of course, over traditional approaches, like Goertzel filters, which are usually slower by an order of magnitude.

You can lower both the MIPS and memory requirements even further by using a staged decimation process. For example, the MFR.1 input signal allows

simple decimation-by-2 filtering, since the highest frequency is sufficiently offset from 2 kHz for the filters to be organized in a way that makes half of the coefficients equal to zero. In addition, the filters in the second stage become half as long, also lowering program RAM requirements.

MANY APPLICATIONS

You can use very similar techniques for different applications: almost any kind of tone detector, signal classifier, or spectrum analyzer and a wide variety of filter banks, can achieve better performance with this approach.

But how much that performance gain benefits the end user depends on many factors. Although you can improve the performance of an entire signal detection and classification subsystem significantly with these techniques, the subsystem usually represents only a small part of the whole picture. If an entire voice processing system is built using a so-called "universal port"



Don't Panic... Toro is your amigo!

Grab difficult data acquisition applications by the horns with Toro, our powerful new 64-bit PCI, TMS320C6711 DSP board with 16 A/Ds & 16 D/As.

Features

- ▶ Ultra-Flexible Triggering Modes
- ▶ Blazingly Fast 64-bit PCI Bus
- ▶ 150 MHz TMS320C6711 DSP (floating point) with 32MB onboard RAM
- ▶ 16 Independent Noise-Free 16-bit, 200 kHz Analog Input & Output Channels
- ▶ Multiboard Synchronization
- ▶ 64 Bits Digital I/O

Applications

- ▶ High Channel Count Servo Controller
- ▶ Vibro-Acoustic Monitoring & Control
- ▶ Sonar
- ▶ Noise Cancellation
- ▶ Beam Forming



**Innovative
Integration**
— real time solutions —

805.520.3300 phone
www.innovative-dsp.com

approach—one in which a single DSP performs all the functions required for one or few channels—the benefits will be modest because the savings in required processor power are minor relative to a typical vocoder's requirements. If, on the other hand, the voice processing is distributed—that is, if various DSPs are assigned to different functions—and each DSP executes only a few smoothly coexisting algorithms, the difference can be dramatic. There are three main reasons.

For one, such a system may be composed of the best available interoperable algorithms from different vendors. For another, fewer DSPs will likely be needed to achieve the same channel density, since every algorithm may trade off between program memory and per-channel performance. Finally, there'll likely be no need to attach any external RAM to the DSPs because the DSPs won't have to keep the entire set of algorithms in their program space. As an added bonus, power consumption will be reduced to just the very low C55x core power plus serial peripherals.

Taken together, these factors will reduce the overall system size and price of future telecommunication infrastructure equipment while increasing its performance, quality, and reliability. ♦

NOTES

[1] R. G. Vaughan, N. L. Scott, and R. D. White, "The Theory of Bandpass Sampling," *IEEE Transactions in Signal Processing*, vol. 39, no. 9 (September 1991), pp. 1973–1985.

[2] S. Kesler, ed., *Modern Spectrum Analysis*, IEEE Press, New York, 1986.

Michael Tsirolnikov (mikets@telus.net) is the principal DSP engineer at MIKET DSP Solutions in Richmond, B.C. He enjoys developing high-quality, high-performance customer-specific applications on the newest DSPs, especially ones presenting significant R&D challenges. His expertise includes system design, adaptive algorithms, and practical implementations of functional transforms.

new

bf3Scp: The ultimate solution for embedded serial communications

- 1 bf3Scp is a serial communications architecture for full-duplex communication between a PC and a DSP.
- 2 bf3Scp enables the rapid development of Windows™ user-interfaces for DSP applications.
- 3 bf3Scp considerably reduces the development time for DSP applications which require serial communications.

Contact us at:
www.windmill-innovations.com

Windmill Innovations

DSP

DSP Solutions

imagine TECHNOLOGY, LLC™

Available Solutions

Audio - MP3, MPEG-2 AAC, MPEG-4 AAC LC Encode/Decode supported

Telephony - G.726, G.711, G.729 AB, G.165/168, DTMF, Call Progress and Encryption

Speakerphone - AEC, LEC, CNG, VAD, AGC and Noise Cancellation with CVCTM

Wireless Modem - V.22, BPSK, QPSK, FEC and Encryption

CS500™ DSP Platform
C6000™ DSP Platform
OMA™ DSP Platform
DSC™ DSP Platform

Imagine Technology
4711 Innovation Drive, Suite # 110
Lincoln, NE 68521 USA
Tel: 402-472-3321 Fax: 208-545-7811
sales@imaginetechology.net
www.imaginetechology.net

iNgress DSP Compliant

DSP
TEXAS INSTRUMENTS

IMPORTANT NOTICE

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

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

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

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

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

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

Mailing Address:

Texas Instruments
Post Office Box 655303
Dallas, Texas 75265