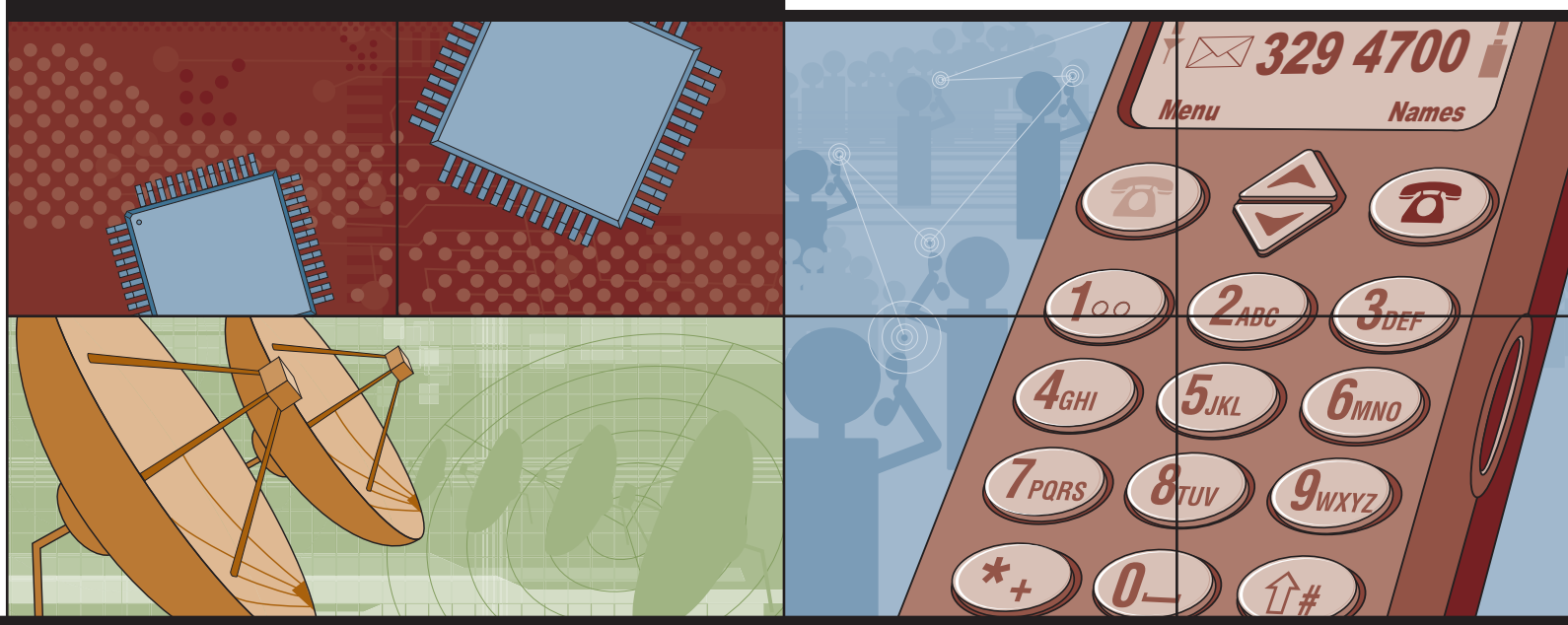
Analog Dialogue

A forum for the exchange of circuits, systems, and software for real-world signal processing



Volume 37, Number 3, 2003

IN THIS ISSUE

ANALOG DEVICES

www.analog.com

Editor's Notes

RF IS EVERYWHERE—BEWARE!

It's high time for a reminder of a theme that cannot be repeated too often: the need for designers of precision low level dc and low frequency equipment to be alert to the adverse effects of ever-increasing ambient high frequency radio-frequency energy, from MHz to GHz, on their measurements. In this world, conductors form antennas, converting electromagnetic waves into voltages and currents—both normal-mode and common-mode. Shielding and filtering are fine, but how many op amp data sheets ordinarily specify CMR at radio frequencies?

We have published warnings. For example, in "Ask the Applications Engineer—14," there is a discussion of high frequency signal contamination,[1] and in a later issue, further illumination of the subject by a reader.[2] Walt Jung's recent book, *Op Amp Applications*, contains substantive discussions—including a useful set of references—in a 30-page section headed "EMI/RFI considerations."[3] You can find an earlier version of this section online in the Hardware Design chapter of the *online* seminar book, *Practical Analog Design Techniques*, edited by Walt Kester.[4] Also, in 1996 (Volume 30, No. 2), we published "A Bibliography on EMC/EMI/ESD," by Daryl Gerke, P.E., and William Kimmel, P.E.,[5] describing a variety of useful texts on these topics.

In *Analog Dialogue* online, 35-4, August 2001, we published an item by a reader, Herman R. Gelbach, P.E., "A Reader Notes," stressing the problem of common-mode RFI. This note has never appeared in print, but it deserves preservation (and our readers need the information), so it is revived below in (relatively) indestructible print.

–***Dan Sheingold,*** **Editor**

A READER NOTES:

[From *Analog Dialogue* 35-4, August 2001. www.analog.com/library/analogDialogue/archives/35-04/reader.html]

High-Frequency-Caused Errors in Millivolt-Measurement Systems

By Herman R. Gelbach, P.E. ret. (hrgelbach@juno.com)

[Editor's Note: Herman never tires of reminding us of the effects of EMI in precision data systems. Faithful readers may recall the adventure that he and our James Bryant shared, summarized in an article that was included in our "Ask The Applications Engineer" collection.[1] *In the present communication, he takes us to task with respect to a couple of recent Analog Dialogue articles, for not once again reminding designers who use precision ICs that they must deal with both normal-mode and common-mode threats to instrumentation-system accuracy. Herman is a Life Fellow of the Instrument Society of America (ISA) and is a design consultant to Scanivalve Co. If you wish to see any of his writings on this subject, get in touch with him at the above email address.]*

I am saddened that the many copies of my ISA paper,[6] *High-Frequency Common Mode, The Contaminator of Signals*, which have been sent to ADI are apparently gathering dust. Scott Wayne states "all Analog Devices instrumentation amplifiers are fully specified for both dc and low-frequency ac common mode rejection."[7] He has completely missed mentioning the source of high-frequency induced errors, unequal slewing of the input device to the impressed signal. I am led to assume that he has never checked for high-frequency common-mode voltages, but a couple of years ago at a seminar that I conducted for the local IES chapter, I checked!

I found in an after-working-hours office environment, on the end of a 50-foot water-pipe-grounded input lead a couple of hundred millivolts of high-frequency CM trash. This will give *significant* offset (slewing) errors in *any* unprotected instrumentation amplifier that I have ever tested—and that's dozens of different commercial instrument designs. See Figure 3 of my paper for a real-life *clean* laboratory environment. It shows about 400 mV p-p of high-frequency trash. Most important, no grounding scheme—except a continuous sheet of copper with *all* system components, including signal wiring and sensors intimately in contact with it—will get rid of this! Maxwell's and Heaviside's equations still are with us. The only practical solution is to prevent the unwanted high-frequency signals from reaching the point of rectification.

A similar comment would apply to Albert O'Grady's article.[8] High-frequency CM is ever present in measurement systems, especially in these days of computers and RF-coupled telephones. Even the design of the remote-sensing transducer excitation-supply error amplifier must consider this error source! On page 37, he talks of parasitic thermocouples and how to eliminate the effect. Unfortunately, such offsets are likely to be insignificant relative to offsets caused by RF induced in the system's wiring. The application of the suggested process only contributes more unknown errors. Gold-copper and copper-copper thermocouples have an extremely low output, so the source of the errant emf is not thermoelectric if normal care is used in the system design.

A test that I had run many years ago tested the variation in voltage at an amplifier input from a loop consisting of signal-conditioner board-edge connector, AMP patchboard connectors, balance pit patch board Deutsch connectors, in-model Winchester SMRE connectors, and was terminated in the two wires being connected to the two terminals of a Constantan strain gage glued to the model structure. Copper-Constantan thermocouples have a very high output; therefore an output might be expected if the temperature of the couples was not exactly matched. The total loop length was perhaps 200 ft.

The loop voltage was observed with an Astrodata Nanovoltmeter, easily capable of 100-nanovolt stability over the test time. The observed loop-voltage variation over the 8-hour shift was 3 microvolts, p-p. This included wind-tunnel warmup and several Mach series from 0.3 to 0.95 Mach. Balance pit and model temperatures, with the several connectors, varied to 130°F. In another test of a few Dynamics, Inc., amplifiers, they were found to repeat offsets within 1/8 microvolt RTI over a week's time using the normal system calibration relays and resistance dividers. *These are raw data without any modification*. The amplifiers have PMI/ADI MAT01 matched monolithic dual-transistor input pairs. Because of the results of the two cited tests, I am completely unimpressed with discussions of correcting for thermal offsets in wiring.

If any of your readers (or colleagues) are interested in my comments, and can't find a copy of the above-mentioned ISA paper, or if they have an interest in a writeup that I made for Scanivalve Co.,[9] "What do I do with this third wire?" please email me. The latter article should be required reading for any one interested in the subject of "grounding" and shielding of sensors, and their interconnection to the receiving device and its "ground."

(continued on page 14)

How to Save Power in Battery Applications Using the Power-Down Mode in an ADC

By Mercedes Casamayor
[mercedes.casamayor@analog.com]
Claire Croke [claire.croke@analog.com]

Size and power consumption are two critical features in portable battery-powered applications. Otherwise acceptable components can be designed out of portable systems based on deficiencies in these two features alone. Everybody desires smaller, more compact mobile phones, MP3 players, PDAs, and digital cameras—with increased time between battery charges or replacement. For semiconductor manufacturers, this translates into a requirement for lower power ICs with high performance and the same—or even extra—features in ever smaller packages.

In portable battery-powered applications, battery life is a critical concern to the system designer. Battery discharge curves differ, depending on the type of battery and the current drain. For example, Figure 1 shows the typical discharge curves for a Lithium/MnO_2 (primary) cell with three typical current loads. They show that the higher the current it must supply, the shorter the battery's life. Since even small amounts of current shorten the battery's life, minimizing the current drawn quiescently by the system components when not operating—or whenever possible during operation—can extend battery life.

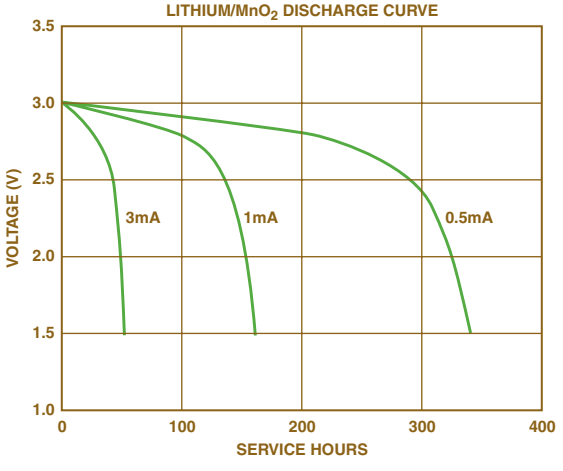


Figure 1. Typical discharge curves.

Nowadays, almost every analog/digital converter (ADC) sold into the battery-powered device market provides a power-down mode as a standard feature. The technique used to place the ADC into the power-down state—and its effectiveness—differ from part to part.

Some ADCs have a dedicated shut-down pin to shift the device into power-down mode. The weakness of this approach is that an extra pin, which results in increased pin count for the ADC, can increase the package size. Other ADCs, like the AD7887, require a *write* to an on-board control register to produce a power-down state. This is generally the case with multichannel ADCs, where an internal register is used for channel selection as well as mode selection. This on-board register also means an extra DATA IN serial interface pin.

In order to cut down on pin count, some recent ADCs use the standard interface lines to implement power-down modes; an example is the 12-bit, 1-MSPS AD7476A, available in the tiny 6-pin SC70 package.

The AD7476A's 3-wire read-only serial interface not only controls the conversion process and accesses the conversion result from the ADC—it is also used to establish the device's different operating modes. The mode of operation is selected by controlling the state of $\overline{CS}$ (*conversion start*) during a conversion. This has the advantage that the signals required to change modes are standard serial interface signals.

The serial interface consists of the $\overline{CS}$, SCLK, and SDATA lines. A normal conversion requires sixteen serial clock pulses for completion. The $\overline{CS}$ signal is used to initiate the conversion and to frame the sixteen serial clocks. After the conversion has been initiated, the time at which $\overline{CS}$ is pulled *high* will determine if the AD7476A will enter power-down mode—or, if already in a power-down mode, whether or not the AD7476A will return to normal operation. Changing the mode of operation can easily be done with a standard 8- or 16-pulse SCLK burst from a microcontroller—or with a framing signal of any length from a DSP.

Figure 2 shows the timing diagram during a normal conversion, and Figure 3 shows how the power-down mode can be entered by controlling the $\overline{CS}$ signal. This mode of operation is designed to provide flexible power management options and to minimize power dissipation for different application requirements.

To reduce power consumption and maintain battery life, the AD7476A should be placed into its low power state between conversions or after a burst of several conversions.

Figure 2. Serial interface signals in a normal conversion.

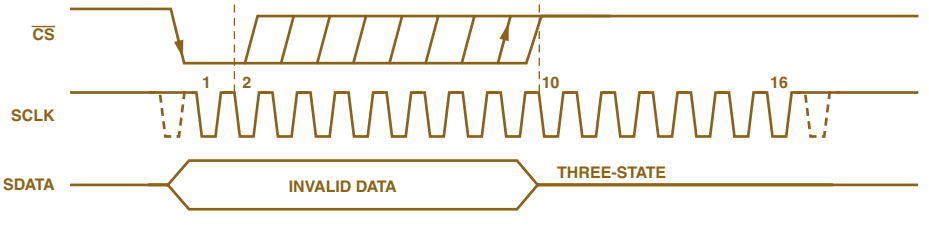


Figure 3. Using the serial interface signals to enter power-down mode.

More about the AD7476A

The AD7476A is a 12-bit successive approximation (SAR-type) ADC, operating on a 2.35-V to 5.25-V supply and capable of throughput rates of up to 1 MSPS. The AD7476A combines CMOS technology and advanced design techniques to achieve low power-dissipation at high throughput rates.

The AD7476A's average power consumption during the cycle time is determined by the percentage of time it spends in a full power state (operational), as compared to the interval spent in a low power state (power down). The greater the time spent in power-down, the lower the average power consumption.

To achieve the lowest power dissipation with the AD7476A, the conversion should be run as quickly as possible. Since the conversion time is determined by the SCLK frequency, the faster the SCLK frequency, the shorter the conversion time. Thus, the device can remain in the power-down mode for a longer interval and will dissipate maximum power for a shorter time.

Figure 4 shows the average power consumption by the AD7476A for different SCLK frequencies with a fixed throughput rate of 100 kSPS. The ADC is put in the power-down mode after the conversion is complete, and is powered up by means of a dummy conversion. As the plot shows, the faster the clock frequency, the lower the average power consumption.

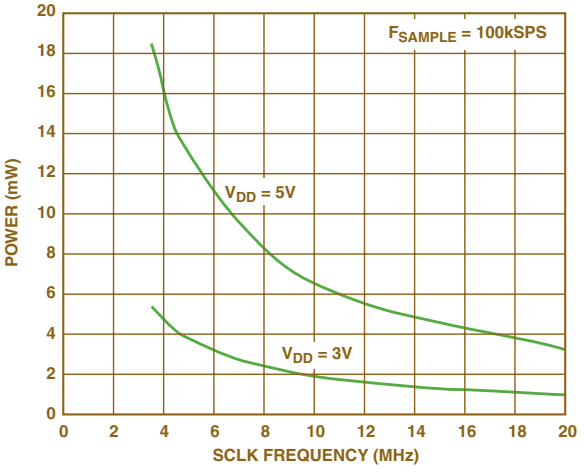


Figure 4. AD7476A power consumption for different serial clock frequencies.

Figure 5 shows that for a fixed SCLK frequency of 20 MHz, when operating the ADC at low throughput rates, the average power consumed by the ADC is very low. However, as the throughput rate increases, the average power consumption increases, because the ADC remains in a power-down state for a shorter period of time compared to the time in the operating state. The other plot shows the average power consumed by the ADC when there is no power-down mode implemented between conversions. Together they show that—while at lower throughput rates significant power savings can be achieved by placing the ADC into a power-down state between conversions—increasingly diminished power savings accrue as the conversion rate increases. For example, at 300 kSPS, the difference between the two cases is less than 0.5 mW.

A further step in the different power-down modes implemented through standard serial interface signals is the *automatic* power-down mode. Following the trend of very low power ADCs for portable battery-powered applications, Analog Devices has recently made available the AD7466, a *micropower*, 12-bit SAR-type ADC housed in a 6-lead SOT-23 package. It can be operated from 1.6 V to 3.6 V and is capable of throughput rates of up to 200 kSPS.

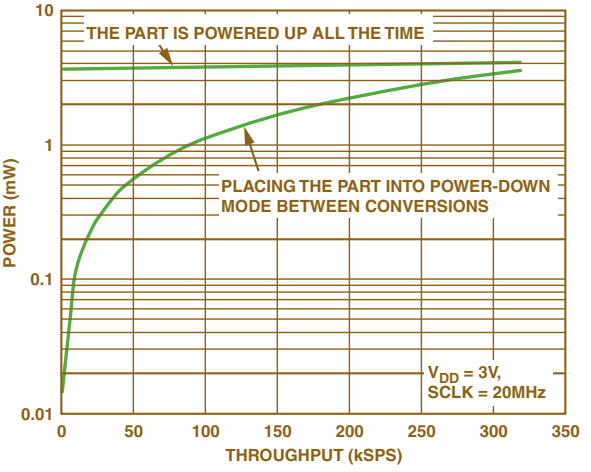


Figure 5. AD7476A power consumption comparison.

The AD7466 powers up prior to conversion and returns to power-down mode when the conversion is complete; this eliminates the need for dummy conversions. In the same way as for the AD7476A, the AD7466's conversion time is determined by SCLK, allowing the conversion time to be reduced by increasing the serial clock speed, thus providing the same kind of power saving.

Figure 6 shows the AD7466's power consumption for different throughput rates, serial clock frequencies, and supplies. The current consumption in power-down mode is typically 8 nA. The AD7466 consumes 0.9 mW max when operating at 3 V, and 0.3 mW max for 1.8 V operation at 100 kSPS.

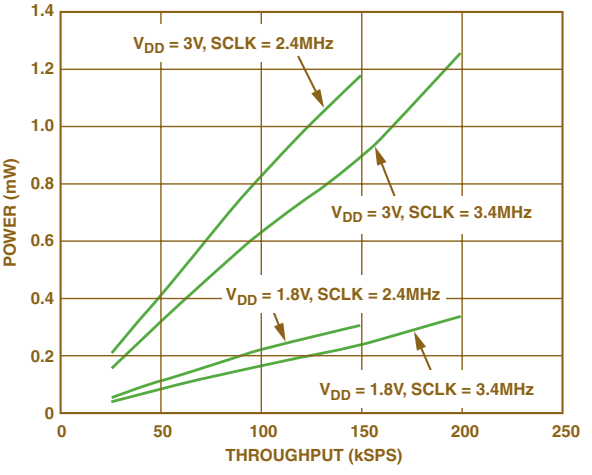


Figure 6. AD7466 power consumption vs. throughput rate for different SCLK and supply voltages.

We have shown that faster SCLK frequencies and longer power-down modes greatly reduce the average power consumed by the ADC. These power savings, combined with the space-saving 6-lead 2 mm × 2.1 mm SC70 surface-mount package, make the AD7476A an ideal candidate for portable battery-powered applications and a very compact alternative to other solutions. And for extremely low-power-budget applications powered at ≤3.6 V, the AD7466 is the ideal solution.

Advanced Digital Post-Processing Techniques Enhance Performance in Time-Interleaved ADC Systems

By Mark Looney [mark.looney@analog.com]

INTRODUCTION

Time interleaving of multiple analog-to-digital converters by multiplexing the outputs of (for example) a pair of converters at a doubled sampling rate is by now a mature concept—first introduced by Black and Hodges in 1980.[1, 2] While designing a 7-bit, 4-MHz A/D converter (ADC), they determined that a time-interleaved solution would require less die area than a comparable 2^n flash converter design. This new concept proved of great value in their design, but saving space was not its only benefit. Time interleaving of ADCs offers a conceptually simple method for multiplying the sample rate of existing high-performing ADCs, such as the 14-bit, 105-MSPS AD6645 and the 12-bit, 210-MSPS AD9430. In many different applications, this concept has been leveraged to benefit systems that require very high sample rate analog-to-digital conversion.

While the speed and resolution of standard ADC products have advanced well beyond 4 MHz and 7 bits, time-interleaved ADC systems (for good reasons) have not advanced far beyond 8-bit resolution. Nevertheless, at 8-bit performance levels, this concept has been widely adopted in the test and measurement industry, particularly for wideband digital oscilloscopes. That it continues to make an impact in this market is evidenced by the 20-GSPS, 8-bit ADC that was recently developed by Agilent Labs[3] and adopted by the Agilent Technologies Infiniium™ oscilloscope family.[4] Indeed, time-interleaved ADC systems thrive at the 8-bit level, but they continue to fall short in applications that require the combination of high resolution, wide bandwidth, and solid dynamic range.

The primary limiting factor in time-interleaved ADC systems at 12- and 14-bit levels is the requirement that the channels be matched. An 8-bit system that provides a dynamic range of 50 dB can tolerate a gain mismatch of 0.25% and a clock-skew error of 5 ps. This level of accuracy can be achieved by traditional methods, such as matching physical channel layouts, using common ADC reference voltages, prescreening devices, and active analog trimming, but at higher resolutions the requirements are much tighter. Until now devices employing more innovative matching techniques have not been commercially available.

This article will outline in detail the matching requirements for 12- and 14-bit time-interleaved ADC systems, discuss the idea of advanced digital post-processing techniques as an enabling technology, and introduce a device employing the most promising solution to date, *Advanced Filter Bank* (AFB™), from V Corp Technologies, Inc.[5, 6]

Time Interleaving Process Overview

Time interleaving of ADC systems employs the concept of running m ADCs at a sample rate that is $1/m$ of the overall system sample rate. Each channel is clocked at a phase that enables the system as a whole to sample at equally spaced increments of time, creating the seamless image of a single A/D converter sampling at full speed. Figure 1 illustrates the block- and timing diagrams of a typical four-channel, time-interleaved ADC system. Each of the four ADC channels runs at one-fourth the system's sample rate, spaced at 90° intervals. The final output data stream is created by interleaving all of the individual channel data outputs in the proper sequence (e.g., 1, 2, 3, 4, 1, 2, etc.). In a two-converter example, both ADC channels are clocked at one-half of the overall system's sample rate, and they are 180° out of phase with one another.

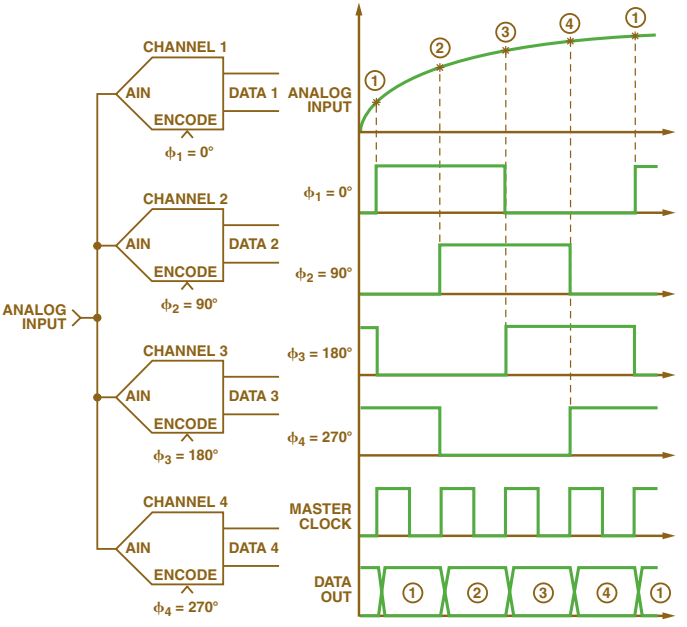


Figure 1. Four-channel time-interleaved ADC system.

For simplicity, this article focuses primarily on two-converter systems, but four-converter systems are discussed when required to articulate key performance differences. Most of the block diagrams, mathematical relationships, and solutions will highlight the two-channel configuration.

Design Challenge of Time Interleaving

As mentioned, channel-to-channel matching has a direct impact on the dynamic range performance of a time-interleaved ADC system. Mismatches between the ADC channels result in dynamic range degradation that—in an FFT plot—show up as spurious frequency components called *image spurs* and *offset spurs*. The image spur(s) associated with time-interleaved ADC systems are a direct result of gain- and phase mismatches between the ADC channels. The gain- and phase errors produce error functions that are orthogonal to one another. Both contribute to the *image-spur* energy at the same frequency location(s). The *offset spur* is generated by offset differences between the ADC channels. Unlike the image spur(s), the offset spurs are not dependent on the input signal. For a given offset mismatch, the offset spur(s) will always be at the same level. Extensive studies of the behavior of these spurs have resulted in several mathematical methods for characterizing the relationship between channel matching errors and dynamic range performance.[7, 8]

While these methods are thorough and very useful, the "error voltage" approach used here provides a simple method for understanding the relationship without requiring a deep study of complex mathematical derivations. This approach is based on the same philosophy used in Analog Devices Application Note AN-501[9] to establish the relationship between aperture jitter and signal-to-noise (SNR) degradation in ADCs. The *error voltage* is defined as the *difference between the "expected" sample voltage and the "actual" sample voltage*. These differences are a result of a large subset of errors that fall into three basic categories: *gain* (Figure 2), *phase* (Figure 3), and *offset* (Figure 4) mismatches.

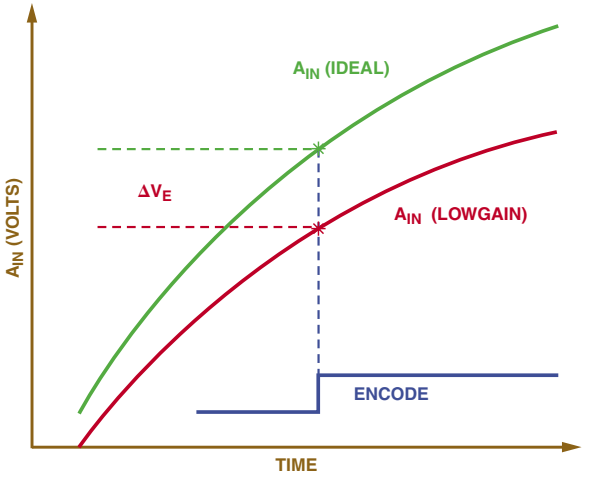


Figure 2. Voltage error due to gain mismatch.

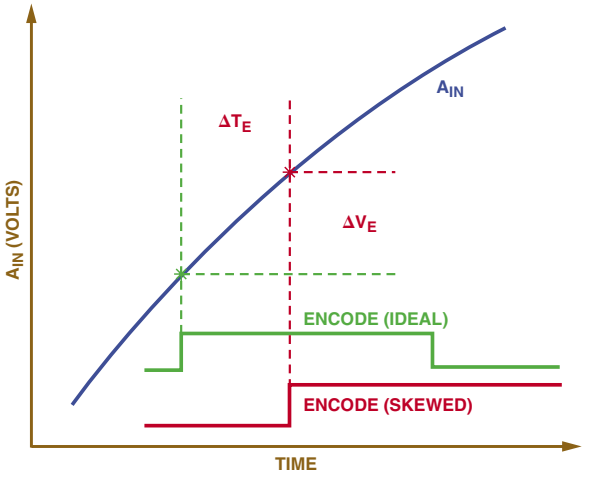


Figure 3. Voltage error due to encode/clock skew.

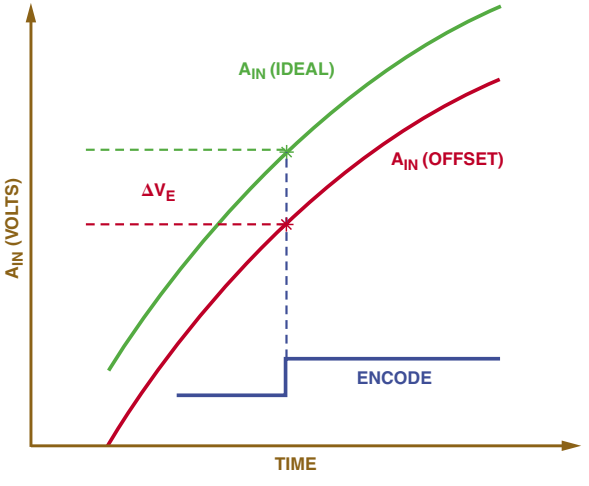


Figure 4. Voltage error due to offset mismatch.

In a two-converter interleaved system, the error voltages generated by *gain* and *phase* mismatches result in an *image* spur that is located at Nyquist minus the analog input frequency. The *offset* mismatch generates an error voltage that results in an *offset spur* that is located at Nyquist. Since the offset spur is located at the edge of the Nyquist band, designers of two-channel systems can typically plan their system frequency around it, and focus their efforts on gain- and phase matching. Figure 5 displays a typical FFT plot for a two-channel system.

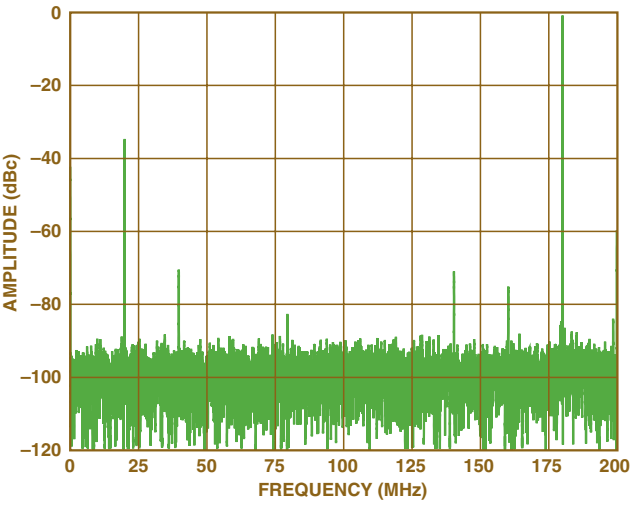


Figure 5. Typical two-converter interleaved FFT plot.

In a four-converter interleaving system, there are three image spurs and two offset spurs. The *image spurs*, generated by gain and phase mismatches between the ADC channels, are located at (1) Nyquist minus the analog input frequency and (2) one-half Nyquist plus or minus the analog input frequency. The *offset spurs* are located at Nyquist and at one-half of Nyquist (middle of the band). Figure 6 displays a typical FFT plot of a four-converter system, illustrating the locations of these five spurs.

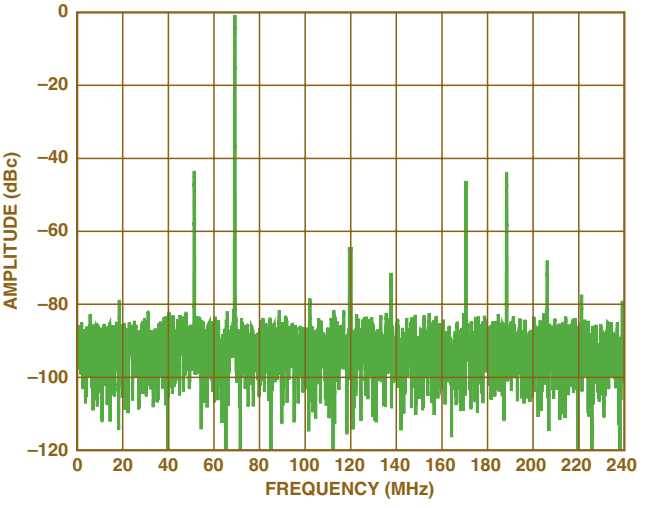


Figure 6. Typical four-converter interleaved FFT plot.

Once the error voltages from each of the three mismatch groups are known, the following equations can be used to calculate the image and offset spurs (IS_{gain}, IS_{phase}, IS_{total}, OS_{offset}) in a single-tone, two-converter system:

$$IS_{gain(dB)} = 20 \times \log(IS_{gain}) = 20 \times \log\left(\frac{G_e}{2}\right)$$
$$where\ G_e = gain\ error\ ratio = \left|1 - \frac{V_{FSA}}{V_{FSB}}\right| \quad (1)$$

$$IS_{phase(dB)} = 20 \times \log(IS_{phase}) = 20 \times \log\left(\frac{\theta_{ep}}{2}\right)$$
$$where\ \theta_{ep} = \omega_a \times \Delta t_e (radians) \quad (2)$$
$$\omega_a = analog\ input\ frequency$$
$$\Delta t_e = clock\ skew\ error$$

$$IS_{total\,(dB)} = 20 \times \log \sqrt{\left(IS_{gain}\right)^2 + \left(IS_{phase}\right)^2} \quad (3)$$

$$OS_{offset\,(dB)} = 20 \times \log\left(\frac{Offset}{2 \times Total\,Codes}\right) \quad (4)$$

$$where\,Offset = channel - to - channel\,offset\,(codes)$$

As noted earlier, the gain- and phase errors generate error functions that are orthogonal[7], requiring a "root-sum-square" combination of their individual contributions to the image spur. Using these equations, an error budget can be developed to determine what level of matching will be required to maintain a given dynamic range requirement. For example, a 12-bit dynamic range requirement of 74 dBc at an input frequency of 180 MHz would require gain matching better than 0.02% and aperture delay matching better than 300 fs! If the gain can be perfectly matched, the aperture delay matching can be "relaxed" to approximately 350 fs. Figure 7 displays an example of a detailed "error budget curve" for this 12-bit, 180-MHz example.

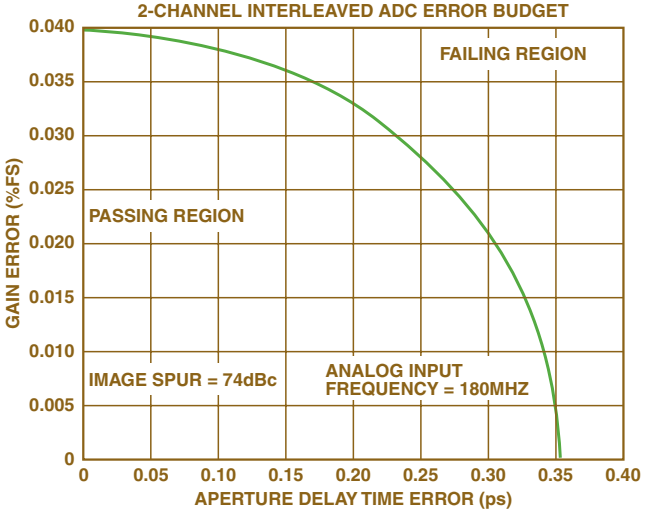


Figure 7. Error budget: 12-bit, 2-channel, 180-MHz input.

Table I provides the matching requirements for several different cases to illustrate the extreme precision required to make a classical time-interleaved A/D conversion system work at 12- and 14-bit resolutions over wide bandwidths.

Table I. Time-interleaved ADC matching requirements.

Performance Requirement at 180 MHz	SFDR (dBc)	Gain Matching (%)	Aperture Matching (fs)
12 Bits	74	0.04	0
12 Bits	74	0	350
12 Bits	74	0.02	300
14 Bits	86	0.01	0
14 Bits	86	0	88
14 Bits	86	0.005	77

Traditional Approach to Wide-Bandwidth Time-Interleaved ADC Systems

A traditional, 2-channel time-interleaving ADC system employs the basic configuration displayed in Figure 8. The first level of matching in traditional time-interleaving ADC systems is achieved through reducing the physical and electrical differences between the channels. For example, gain matching is typically controlled by the use of common reference voltages and carefully matched physical layouts. Phase matching is achieved by manually tuning the electrical length of the clock (or analog input) paths and/or through special trimming techniques that control an electrical characteristic of the clock distribution circuit (rise/fall times, bias levels, trigger level, etc.). The offset matching depends on the offset performance of the individual ADCs.

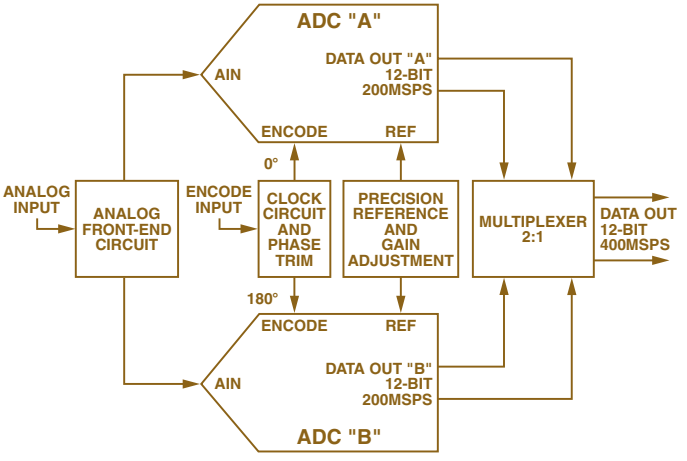


Figure 8. Functional diagram of a traditional time-interleaved ADC.

Many of these matching approaches are based on careful analog design and trim techniques. While there has been an abundance of excellent ideas to address these tough matching requirements, many of them require additional circuits that add error sources of their own—defeating the original purpose of achieving precise gain and phase matching. An example of such an idea would be setting the rise and fall times of the two different clock signals. Any circuit that could provide this level of control would be subjected to increased influence of power-supply voltage—and temperature—on each channel's phase behavior.

Advanced Digital Post Processing

The development of new digital signal processing techniques, along with the advances in inexpensive, high-speed, configurable digital hardware platforms (DSPs, FPGAs, CPLDs, ASICs, etc.), has opened the way for breakthroughs in time-interleaving ADC performance. Digital *post-processing* approaches have several advantages over classical analog matching techniques. They are flexible in their implementation and can be designed for precision well beyond the ADC resolutions of interest. A conceptual view of how digital signal processing techniques can impact time-interleaved system architectures can be found in Figure 9. This concept employs a set of digital calibration transfer functions that process each ADC's output data, creating a new set of "calibrated outputs." These digital calibration transfer functions can be implemented using a variety of digital filter configurations (FIR, IIR, etc.). They can be as simple as trimming the gain of one channel or as complicated as trimming the gain, phase, and offset of each channel over wide bandwidths and temperature ranges.

Wide bandwidth and temperature matching presents the greatest opportunity—and challenge—for using digital post-processing techniques to improve the performance of time-interleaving ADC systems. The mathematical derivations required for designing the digital calibration transfer functions for multiple ADC channels over wide bandwidths and temperature ranges are extremely complex and not readily available. However, a great deal of academic work has been invested in this area, creating a number of interesting solutions. One of these solutions, known as *Advanced Filter Bank* (AFB), stands out in its ability to provide a platform for a significant breakthrough.

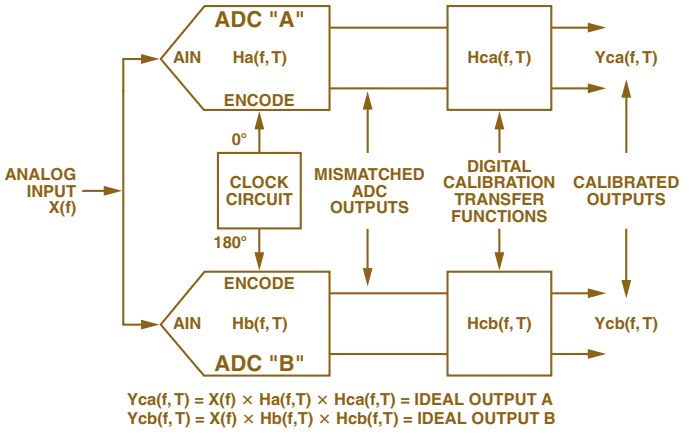


Figure 9. Example of digital post-processing architecture.

Advanced Filter Bank (AFB)

AFB is one of the first commercially available digital post-processing technologies to make a significant impact on the performance of time-interleaving ADC systems. By providing precise channel-to-channel gain, phase, and offset matching over wide bandwidths and temperature ranges, AFB is well-positioned to solidly establish time-interleaving ADC systems in the area of high-speed, 12-/14-bit applications. Besides its matching functions, AFB also provides phase linearization and gain-flatness compensation for ADC systems. Figure 10 displays a basic block diagram representation of a system employing AFB.

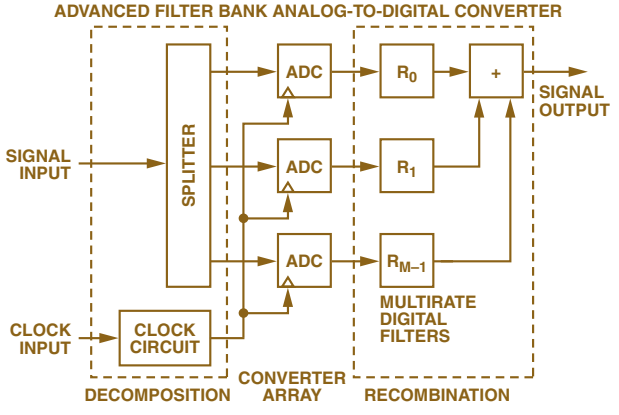


Figure 10. AFB basic block diagram.

By using a unique multirate FIR filter structure, AFB can be easily implemented into a convenient digital hardware platform, such as an FPGA or CPLD. The FIR coefficients are calculated using a patented method that involves starting with the equations seen in Figure 9, and then applying a variety of advanced mathematical techniques to solve for the digital calibration transfer function.

AFB enables time-interleaving ADC systems to use up to 90% of their Nyquist band, and can be configured to operate in any Nyquist zone of the converter (e.g., first, second, third, etc.) The appropriate Nyquist zone can be selected using a set of logic inputs, which control the required FIR coefficients.

AFB Design Example

The AD12400 is the first member of a new family of Analog Devices products that leverage time interleaving and AFB. Its performance will be used to illustrate what can be achieved when state-of-the-art ADC design is combined with advanced digital post-processing technologies. Figure 11 illustrates the AD12400's block diagram and its key circuit functions. The AD12400 employs a unique analog front-end circuit with 400-MHz input bandwidth, two 12-bit, 200-MSPS ADC channels, and an AFB implementation using an advanced *field-programmable gate array* (FPGA). It was designed using many of the classical matching techniques discussed above, together with a very low jitter clock distribution circuit. These key components are combined to develop a 12-bit, 400-MSPS ADC module that performs very well over 90% of the Nyquist band and over an 85°C temperature range. It has an analog input bandwidth of 400 MHz.

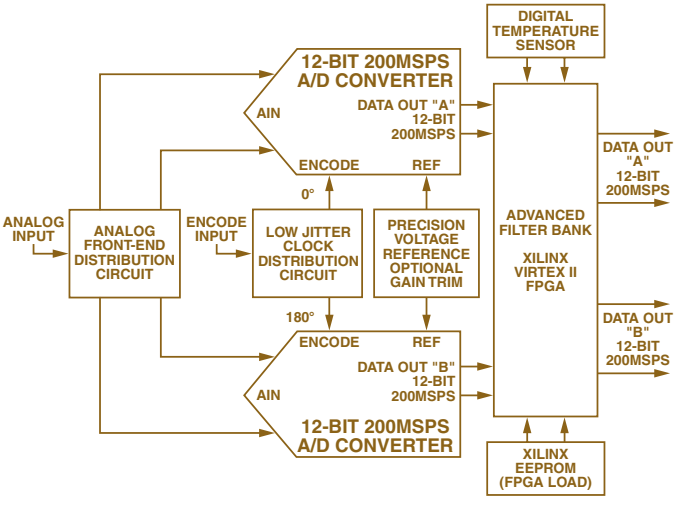


Figure 11. AD12400 block diagram.

The ADCs' transfer functions are obtained using wide-bandwidth, wide-temperature range measurements during the manufacturing process. This characterization routine feeds the ADCs' measured transfer functions directly into the AFB coefficient calculation process. Once the ADCs have been characterized, and the required FIR coefficients have been calculated, the FPGA is programmed and the product is ready for action. Wide bandwidth matching is achieved using AFB's special FIR structure and coefficient calculation process. Wide temperature performance is achieved by selecting one of the multiple FIR coefficient sets, using an on-board digital temperature sensor.

The true impact of this technology can be seen in Figures 12 and 13. Figure 12 displays the *image-spur* performance across the first Nyquist zone of this system. The first curve in Figure 12 represents the performance of a 2-channel time-interleaved system that has been carefully designed to provide optimal matching in the layout. The behavior of the image spur in this curve makes it obvious that this system was manually trimmed at an analog input frequency of 128 MHz. A similar observation of Figure 13 suggests a manual trim temperature of 40°C.

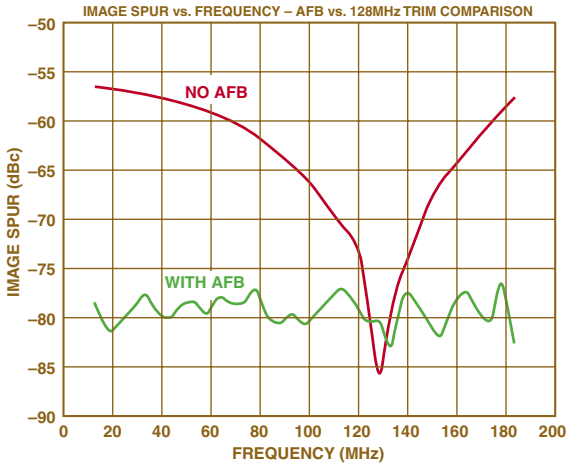


Figure 12. Performance of a manually trimmed system "before and after" AFB compensation over the frequency range.

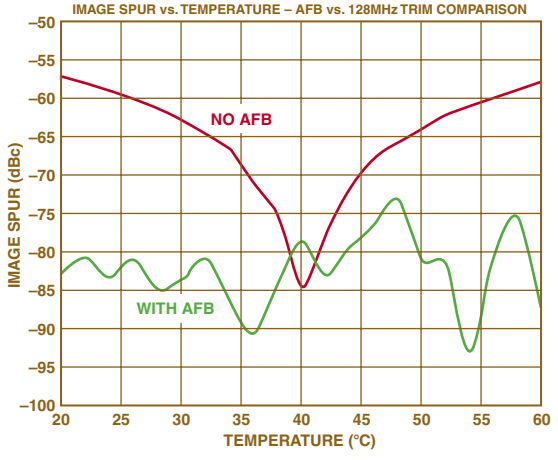


Figure 13. Performance of a manually trimmed system "before and after" AFB compensation over the temperature range.

Despite a careful PCB layout, tightly matched front-end circuit, tightly matched clock-distribution circuit, and common reference voltages used in the AD12400 ADC, the dynamic range degrades rapidly as the frequency and/or temperature deviates from the manual trim conditions. This rapid rate of degradation can be anticipated in any two-converter time-interleaved ADC system by analyzing some of the sensitive factors affecting this circuit. For example, the gain-temperature coefficient of a typical high-performance, 12-bit ADC is 0.02%/°C. In this case, a 10°C change in temperature would cause a 0.2% change in gain, resulting in an image spur of 60 dBc (see Equation 1). Considering just this single ADC temperature characteristic, the predicted image spur is 3 dB worse than the 30°C performance displayed in Figure 13.

By contrast, the dynamic range performance shown in these figures remains solid when the AFB compensation is enabled. In fact, the dynamic range performance surpasses the 12-bit level across a bandwidth of nearly 190 MHz and a temperature range of 40°C. Another significant advantage of this approach is that the temperature range can actually be expanded from the 20°C to 60°C range shown to 0°C to 85°C by using additional FIR coefficient sets—as embodied in the AD12400.

CONCLUSION

Time interleaving is growing into a significant trend in performance enhancement for high-speed ADC systems. Advanced digital post-processing methods, such as AFB, provide a convenient solution to the tough channel-matching requirements at resolution levels that were not previously achievable for time-interleaved systems. When combined with the best ADC architectures available, advanced DSP technologies, such as AFB, are ready to take high-speed ADC systems to the next level of performance and facilitate greatly improved products and systems in demanding markets such as medical imaging, precise medicine dispensers (fluid flow measurement), synthetic aperture radar, digital beam-forming communication systems, and advanced test/measurement systems. This technology will result in many breakthroughs that will include 14-bit/400-MSPS and 12-bit/800-MSPS ADC systems in the near future.

ACKNOWLEDGEMENT

The author would like to thank Jim Hand and Joe Bergeron for their guidance and innovative insights offered for this article. Their contributions are greatly appreciated.

REFERENCES

[1] W. C. Black Jr. and D. A. Hodges, "Time Interleaved Converter Arrays," *IEEE International Conference on Solid State Circuits*, Feb 1980, pp. 14–15.

[2] W. C. Black Jr. and D. A. Hodges, "Time Interleaved Converter Arrays," *IEEE Journal of Solid State Circuits*, Dec 1980, Volume 15, pp. 1022–1029.

[3] K. Poulton, et al., "A 20GS/s 8-b ADC with a 1MB Memory in 0.18µm CMOS," *IEEE International Conference on Solid State Circuits*, Feb 2003, pp. 318-319, 496.

[4] Press Release, "Agilent Technologies introduces industry first 6-GHz, 20-GSa/s-per-channel oscilloscope and probing measurement system," Agilent Technologies Web Page, Nov 1, 2002, http://www.agilent.com/about/newsroom/presrel/archive.html

[5] S. Velazquez, "High-performance advanced filter bank analog-to-digital converter for universal RF receivers," *IEEE SP International Symposium on Time-Frequency and Time-Scale Analysis*, 1998, pp. 229–232.

[6] Technical Description, "Advanced Filter Bank (AFB) Analog-to-Digital Converter Technical Description," V Corp Technologies, http://www.v-corp.com/analogfilterbank.htm

[7] N. Kurosawa, et al., "Explicit Analysis of Channel Mismatch Effects in Time Interleaved ADC Systems," *IEEE Transactions on Circuits and Systems I—Fundamental Theory and Applications*, Volume 48, Number 3, Mar 2003.

[8] M. Gustavsson, J. J. Wikner and N. N. Tan, *CMOS Data Converters for Communications*, Boston: Kluwer Academic Publishers, 2000, pp. 257–267.

[9] B. Brannon, "Aperture Uncertainty and ADC System Performance," Analog Devices, Inc. Application Note, AN-501, http://www.analog.com/UploadedFiles/Application_Notes/365163734AN501.pdf

Dynamic Memory Allocation Optimizes Integration of Blackfin® Processor Software

By Lidwine Martinot [lidwine.martinot@analog.com]

Typical DSPs usually have a small amount of fast on-chip memory. Microcontrollers usually have access to larger external memories. The Blackfin processor has a hierarchical memory architecture that combines the best of both approaches, providing several levels of memory with differing performance levels. For applications that require the most determinism, it can access on-chip SRAM in a single core clock cycle. For systems that have larger code sizes, larger on-chip and off-chip memory is available—with increased latency.

By itself, this hierarchy is only moderately useful; today's high-speed processors would typically run at much slower speeds, because larger applications would only fit in slower external memory. To improve performance, programmers have the option of manually moving key code in and out of internal SRAM. Also, the addition of data and instruction caches into the architecture makes external memory much more manageable. The cache reduces the manual movement of instructions and data into the processor core. This greatly simplifies the programming model by eliminating the need to worry about managing the flow of data and instructions into the core.

While Blackfin's memory is versatile and easy to use in many applications, there are some applications, such as embedded cell phone systems, in which memory allocation can be difficult for *any* embedded processor. In this kind of application, the instruction cache does not provide the same level of code management as manual movement of data in and out of SRAM. This article suggests a dynamic memory allocation tool to deal with the challenge.

An essential element in the development of protocol stack and application software for mobile phone platforms is the efficient handling of memory resources in the system. In the past, memory resources were distributed "by hand" to each piece of code within the system; but the growing number of modules such as video and voice recognition makes solutions using this approach more challenging to optimize. A *dynamic memory allocator* can be used to allocate and free memory in a large application, removing the need to manage this task manually. This article describes some of the principles of dynamic memory allocation and demonstrates a specific implementation that takes into account the overall system considerations and the division of Blackfin's memory into different spaces with various properties (price, speed, dual-access possibility).

Memory management solutions

In a large embedded application, there are several memory-management approaches that can be realized. The major approaches are described below.

Stack. All variables and buffers can be simply declared on top of a function. They are stored in the Stack space, and that space is released only when exiting the function. The main disadvantage of this solution is Stack growth, e.g., the Stack keeps growing during the function's lifetime. Its lifetime can sometimes be very long, since the function may be recursive and/or interruptible.

Manual overlap. Another popular solution consists of hard-coding the buffer's address using sections defined at the *link* stage. This is a bit more flexible than allocating in the stack, because it allows memory overlap. If two modules are never going to interrupt each other, their temporary memory could share the same memory section. Yet, as the number of modules grows, this solution really becomes difficult to manage for an integrated system. In addition, other memory problems—such as inappropriate overlap, or insufficient buffer sizes for a given section—can be very hard to track. To make matters worse, it is even more difficult when a new feature is needed that requires two functions that have never previously overlapped in time to run concurrently. Figure 1 shows an example of a manual overlap-based implementation.

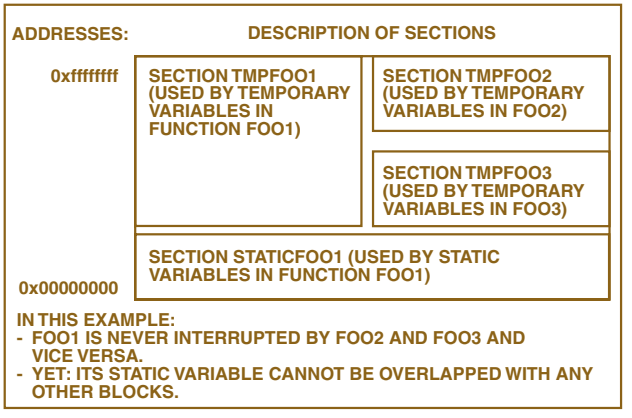


Figure 1. Manual overlap of memory.

Dynamic allocation. Dynamic allocation enables *memory overlap*: once a memory space is not needed, it is freed and can be reused. Unlike the stack allocation method, dynamic allocation does not result in an increase of uncontrolled memory space. In fact, the memory used by a function is released as soon as it is not required, rather than waiting for the end of the function.

What are the features to consider when developing a dynamic memory allocator?

A dynamic memory allocator is made up of two functions: one allocates memory space; the other frees memory. The allocation reserves some space to serve memory requests. When the *free* function has been called, the reserved space is freed and can be used to fulfill further requests. For example, let's build a very basic dynamic memory allocator to understand all the trade-offs such a piece of code has to deal with. We will start with some basic definitions and then describe the allocator.

Chunk. Let's assume the allocator can give the required memory a *chunk* of a big memory space. It is easy to understand that the whole space cannot be taken away to serve the first request. Instead, the initial memory space can be split into different chunks of different sizes.

Header. When a memory request is made, how do we know that a given piece is big enough? The *size* has to be kept in memory somewhere. One solution among others is to keep it in a *header* inside the chunk. This is an element of *memory overhead.* Also, at least one bit in the header needs to be dedicated to indicate whether the chunk is free or is in use.

Wandering through the chunks. If the first chunk is too small, how do we jump to the next chunk? If all chunks are consecutive in memory, it is enough to know the size of the chunk to jump to the next. Another solution consists of keeping a *pointer* to the next chunk in the header—this is the principle of *linked lists.*

Finding a fit. How do we select which free block is going to serve the request? A necessary condition is to find a free chunk whose size is at least the required size. The first chunk that meets this requirement can then be used. This policy is called the *first-fit*. Another policy, the *best-fit* policy, consists of looking for the *smallest free chunk* that can accommodate the request. This is the most challenging dilemma a dynamic memory allocator has to deal with: *speed versus memory size*. The first-fit is fast but might lead to huge memory losses, while the alternative of finding the best fit requires time. A compromise can be reached with the use of several linked lists of chunks (*bins*), in which each list has its chunks of a similar size. The *best-fit* policy selects the bin, while the *first-fit* selects the chunk within the bin.

Fragmentation. Another solution consists of using the first-fit policy—and releasing the end of the chunk that is bigger than the request. One downside of this solution is that soon the memory is made up of several scattered blocks (different in size, usually small) of unused memory. Future allocation is difficult due to the small free spaces that result. This situation is called *memory fragmentation*.

To speed a request, some allocators are based on linked lists of free chunks. This saves some time, since the search can avoid considering all in-use chunks. This method does have a disadvantage, however. If only the free chunks are kept in lists, it is hard to have all of them placed consecutively in memory; this problem prevents the allocator from being able to take two adjacent medium chunks and put them together (or *coalesce* them) to build a bigger one.

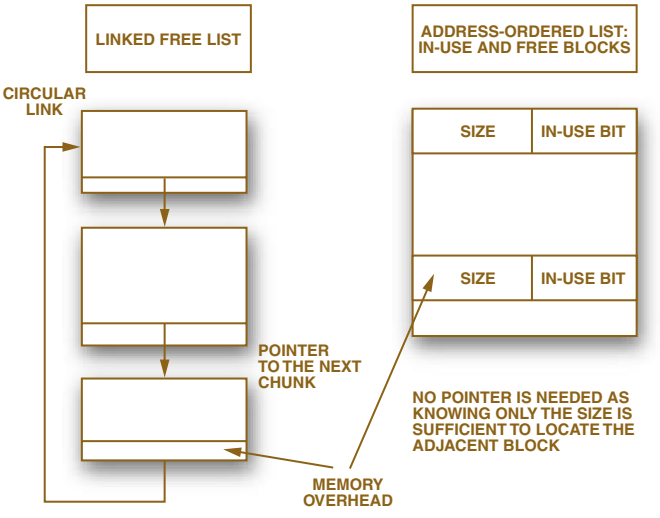


Figure 2. Examples of dynamic allocators.

We have now introduced all the concepts and compromises for understanding the allocator designed for the Blackfin mobile phone system: ADIalloc.

The current implementation: ADIalloc

The constant addition of signal-processing features (new video and audio standards, for instance) has motivated the development of an allocator referred to as ADIalloc for cell phone applications. It is intended to help reduce both *time-to-market* of the product using the processor—by avoiding undesired memory overlap—and *cost*, by reducing the peak memory usage.

Basic principles

The current implementation is more focused on speed performance than memory overhead. The memory is partitioned into *bins*. Each bin holds *chunks* of memory of equal size. The chunks in a bin have consecutive addresses, allowing a fast jump from one chunk to the next. The policy to find the chunk that suits the request is *best-fit* for the bin and *first-fit* within the bin—meaning the first free chunk, since all chunks have the same size. Moreover, the size of chunks in bins is chosen to facilitate finding the best bin: they are all related by powers of 2. Chunks in bin (N+1) are double the size of chunks in bin N (it is also possible for bin N to contain 0 chunks...)

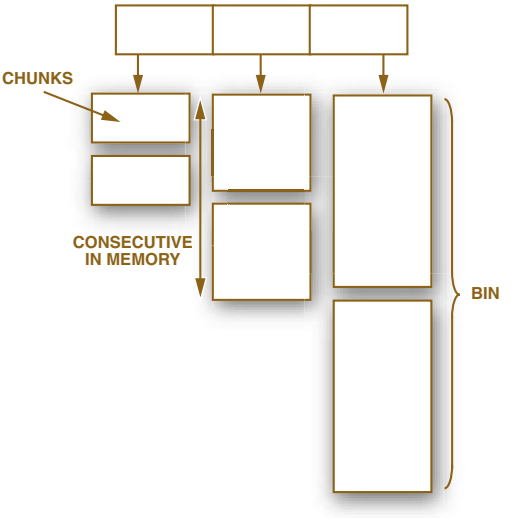


Figure 3. Bins/Chunks configuration of ADIalloc.

Some software modules may occasionally need one "big" chunk. However, if big chunks are allowed, the memory is going to be partitioned into very few chunks. Instead of one big chunk, it is better to have two smaller chunks that would be coalesced together to form a big chunk in the few cases where it is needed. Consequently, *coalescing* two chunks together is allowed.

To guarantee speed, each chunk has a header that indicates if it is available and coalesced. In the case of coalesced chunks, the size of the coalesced companion, or "buddy," is kept in the header. This is used to quickly restore the header of the buddy when the couple is freed.

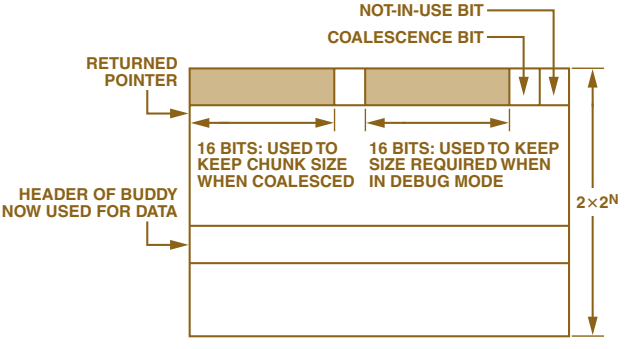


Figure 4. Chunks in ADIalloc.

What is specific to Blackfin

Blackfin adds yet another dimension to the memory allocator: its data memory space is partitioned into several memory levels. The memory levels have different characteristics in terms of price, speed, and dual-access possibility:

- The *external* memory, Lext, is big and less expensive to use—but is accessed with higher latency.
- The *on-chip* memory, L1, has fast access. It is itself split into different banks and sub-banks, allowing two items of data to be accessed at the same time (*dual access*) from separate sub-banks.

- L2 is in between, in terms of price and speed. However its speed can be improved by caching it into L1. *Caching* is an additional dimension.

Stack. Although (as seen earlier) allocating all variables in a Stack is not a good solution, a Stack is still needed. For small buffers, loop counters, and indexes there is no point to losing cycles because of allocation. Yet there might be some uncertainty about the allocation—stack or dynamic—of some buffers until the system-integration stage. This is why the Stack is seen as an additional memory level.

Cache. As mentioned above, Blackfin can cache L2 memory into L1—or parts of L1. In that case, it is advantageous not to have to readapt the allocator's code to the new memory. During initialization, the allocator is able to read the cache configuration from some dedicated Blackfin registers, and then decide about its bins and chunks. Yet since the allocator has to be tested on any platform, it must remain minimally Blackfin-specific. Only reading the data-cache configuration is Blackfin-specific. Apart from that, the allocator can be fully tested on a PC with a compiler other than Blackfin's. The only difference there is that choice of memory resource is not related to the platform's speed or dual-access features.

With all the above features ADIalloc becomes an important piece of software. Therefore it should be made as "flexible" as possible, as long as this does not overly impact the number of cycles.

Flexibility of the allocator

Macro. C-macros are extensively used in the ADIalloc implementation. Indeed *ADIalloc* is itself a macro. The first benefit is to be able to replace quickly one allocator by another without having to rewrite all pieces of software that invoke ADIalloc. For instance, this can be used to investigate the performance of different dynamic allocators.

Alloca. Another advantage of the macro is to be able to use Stack as a memory level without having to invoke the allocator in a more complex manner than would be done with a malloc. Indeed, allocating in Stack cannot be achieved through a function call. Instead, when ADIalloc is invoked with Stack as memory level, 'alloca' is executed. (Alloca is available with most compilers. It reserves space on the Stack only when the alloca instruction is executed—unlike the declaration on the Stack on top of a function, which reserves the space for the function lifetime.) The macro ADIalloc tests the memory level required and redirects it to an alloca or to a function call to the allocator, ADI_alloc.

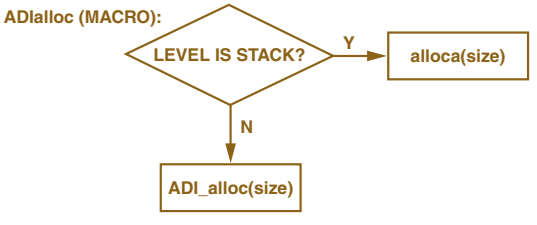


Figure 5. Stack allocation via ADIalloc.

Storage of the desired memory level. It is a really great advantage to be able to deal with the different memory levels on the Blackfin. To make the best use of this feature the memory levels are not fixed at compile time. Hence, for each allocation the allocator allows testing of different memory levels without having to rewrite or recompile the software module's C code. A software module is accompanied by a table that contains the memory level required for such and such allocation. The table's content can be changed at run time as simply as writing a new desired memory level at a specific address. Nevertheless it should be noted that if the memory level required cannot be provided, the allocator picks up another level—the closest one in terms of memory access speed.

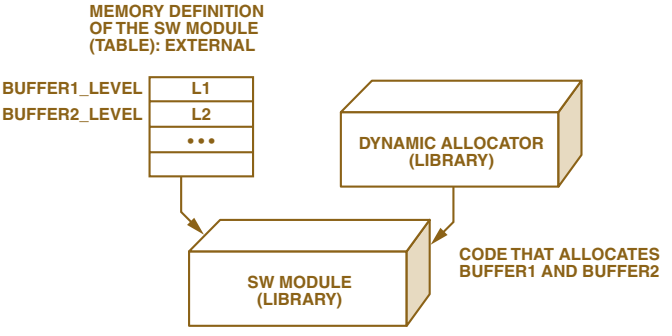


Figure 6. Input table: desired memory level.

Change Bins/Chunks Configuration. Another flexible feature of ADIalloc is the ability to change the bins and chunks configuration without having to recompile the allocator's code. Indeed all variables defining this configuration are saved into tables. The tables are read during the initialization. At any time the tables' content can be changed—which will modify the bins/chunks configuration the next time the initialization is called. Not having to fix the bins/chunks split at compile time leads, as a next feature, to having a smart wrapper around the allocator that dynamically resizes the memory. We can also think of a system running two consecutive tasks that require two different memory configurations. When a task finishes, the allocator initialization is called with the configuration that best suits the second task.

Finally, ADIalloc is derived in two flavors: the first is used for development and integration, the second one is used in the final product. During development *debug* features are mandatory. The next section provides further details of the current implementation and how to make the best use of debug features.

How debug features improve implementation

Common issues when using memory allocator are inefficiencies attributable to the allocator and the risk of not allocating and freeing the memory properly—resulting mainly in memory leakage.

The allocator knows the memory partition. It also knows the amount of memory requested and which memory addresses are free. This allows debug features to be developed to take steps to avoid memory leakage.

Track a free that has been forgotten. The first reason for a memory leak occurs when a memory is allocated but never freed. This can be easily prevented. In debug mode (not in normal mode, since this test takes many cycles) the allocator builds statistics of the memory usage. If the last report shows that some memory space is still in-use, it means a free has been forgotten. To track the problem more deeply, one can use another report which contains buffer names, their addresses, and if they are being freed or allocated (the report is built each time the allocator or the free function is called).

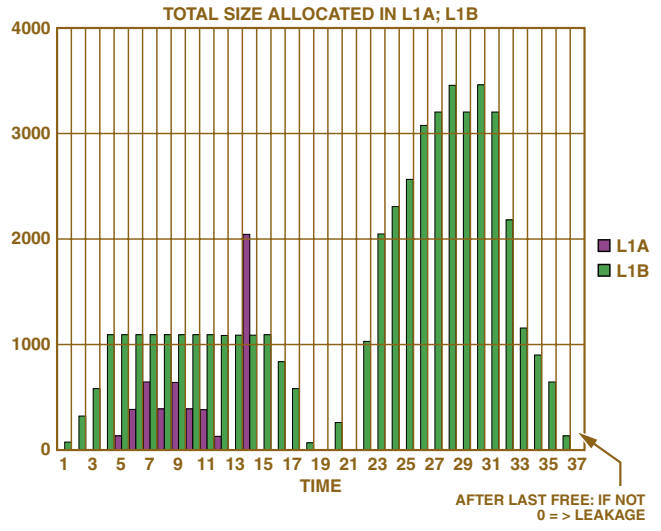


Figure 7. How to track a free that has been forgotten.

Track that more space than reserved is used. The other type of leakage occurs when a buffer allocates less space than what it needs, and starts using the space outside what has been allocated to it. In debug mode the allocator "marks" all free memory spaces with a special code (a code which has a very low probability of being a "real" datum). It not only marks free chunks, but also includes all the addresses inside a chunk not required by the allocation. In each allocated chunk the required size is also kept as part of the allocated chunk. Hence each time the allocator is entered (for a new allocation or a free) it verifies that:

- The free chunks only contain the special code
- The allocated chunks contain the special code between the required size and the end of the chunk

The function that does this check can also be called at any time outside the allocator. When leakage is noticed, a message is built and passed to another module, which outputs it in one form or another (screen, special visualization tool, high-speed logger for real-time analysis, etc.)

```
ADIalloc:ptrB   adi_free_log_when_leak_t xx;
ADIfree: ptrB   xx.address_freed       = (void*)  0x00429374;
                xx.error_warning       = ERROR_PROBABLY_LEAKAGE;
                xx.address_of_leakage = (void*)  0x004291B4;
```

Figure 8. Example of a viewer to track the allocator messages (case of leakage).

Help select bins/chunks configuration. The allocator debug features can also partly resolve the concerns regarding the allocator inefficiencies. In debug mode the allocator saves such data as the memory required versus the memory allocated, the number of chunks used per bin, etc. This provides an easy way to avoid big inefficiencies—such as having some bin sizes that are never used.

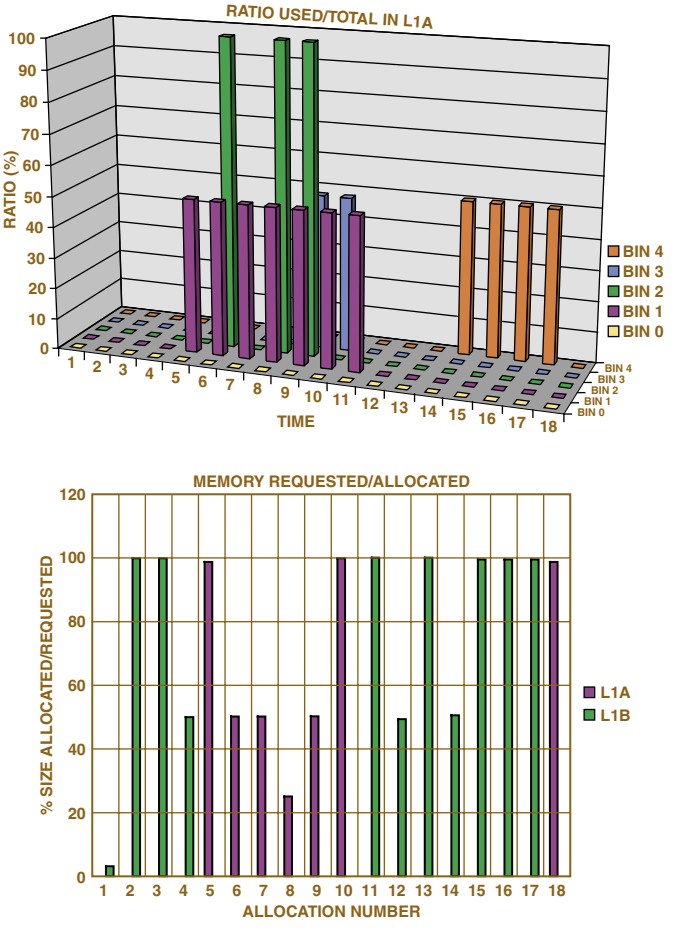


Figure 9. Data captured to help select the best Bins/Chunks configuration.

Memory repartition between memory levels. A big concern is then how to apportion the memory levels among the different pieces of software. Obviously, the fast-access memory suits best every single piece of code. Yet a choice has to be made since this memory is limited. This choice can be made only once whole software modules are built into a *system*. Usually the time-critical tasks will need the fastest memory. The allocator can assist in making such choices.

The allocator is all the more helpful, as it can be delivered with a wrapper that takes care of running all possible memory configurations for a specific module while conserving the number of cycles required. This helps one to know the impact on cycles of not being able to get the fastest memory for a specific buffer.

Table I. Performance Matrix

Index In Table	L1_B	PASS/FAIL	L2	PASS/FAIL	Lext	PASS/FAIL
pChannelInstance	–82	PASS	–71	PASS	–119	PASS
pSharedMemStruct	–73	PASS	–66	PASS	–109	PASS
pShared_BurstDec_CCDec_Interleave	94	PASS	56	PASS	–48	PASS
pShared_EQ_CCDec_Mod_Info	5	PASS	–81	PASS	–67	PASS
CC_Dec_IO_EDGE_PDTCH	130*	PASS	–74	PASS	324	PASS
pDeInterleave	–232	PASS	–57	PASS	18115	PASS
pOutHeader	15	PASS	–116	PASS	506	PASS
pScratch_Header_Decoder	–281	PASS	–83	PASS	3719	PASS
Metric	–82	PASS	10440	PASS	123346	PASS
pPathMetric	–417	PASS	–84	PASS	77394	PASS
pOutRLC_Data	–199	PASS	–83	PASS	1832	PASS
pScratch_Data_Decoder	–75	PASS	450	PASS	23624	PASS

*Means: +130 cycles if the buffer is in L1_B compared to the reference configuration.

The numbers shown in the table represent the difference in the number of cycles needed to run the unit test in the new configuration as compared to the reference configuration.
The Reference Configuration is what is provided as default by the module's writer.
PASS indicates that the result of running the unit test on the new configuration is the same as that of running the reference configuration.
The Reference Number of cycles is: 128078.

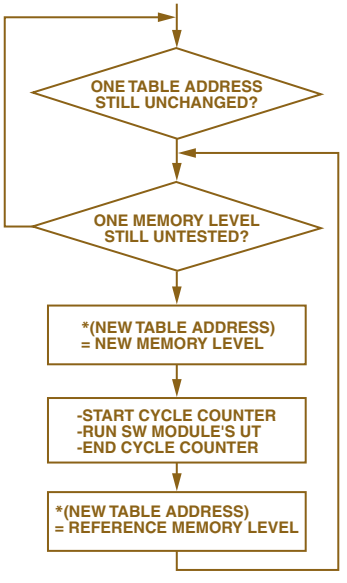


Figure 10. Unit test flowchart.

The wrapper runs a software module *unit test* (UT). The first time it runs it, the allocator is asked to return the pointer's name and the address of the table where it looks for the memory level. After collecting all addresses where it needs to look for memory levels, the wrapper re-runs the UT for all possible memory configurations.

CONCLUSION

The current ADIalloc implementation is one possible implementation of a dynamic memory allocator. Its use has shown that the most useful features of the current implementation are the *debug* features. They reduce the risks linked to dynamic allocation (especially the risks of leakage). At the same time they help better manage complex memory structures. It has now become much easier in cell phone applications to add new software modules inside Blackfin without having to rework the division of memory between modules.

ACKNOWLEDGEMENT

The author acknowledges the invaluable contributions of Rick Gentile, of the Blackfin Applications Group, and Zoran Zvonar, DSP/Systems group leader.

(continued from page 2)

Footnotes

[1] *Analog Dialogue* 27-2, 1993, pp. 28-29. http://www.analog.com/library/analogDialogue/Anniversary/14.html

[2] *Analog Dialogue* 28-2, 1994, p. 23. http://www.analog.com/library/analogDialogue/Anniversary/Reader_Notes.html

[3] Jung, Walter G., editor, *Op Amp Applications*. Norwood, MA: Analog Devices, Inc., ©2002, pp. 7.109 to 7.138. To purchase a copy of the book, phone 1-800-262-5643 (North America). (Code OP-AMP-APPLIC-BOOK, $40). Outside North America, call your nearest ADI sales office.

[4] Kester, Walt, editor, *Practical Analog Design Techniques*. Norwood, MA: Analog Devices, Inc., ©1995. "EMI/RFI considerations" can be found at http://www.analog.com/UploadedFiles/Associated_Docs/53333534294954127648933Section9.pdf (pages 51 through 79).

[5] Gerke, Daryl, P.E., and William Kimmel, P.E., "A Bibliography on EMC/EMI/ESD and other threats to signal and circuit integrity." *Analog Dialogue* 30-2 (1996), p. 11. Also available online at www.analog.com/library/analogDialogue/archives/30-2/bibliography.html

[6] Gelbach, Herman, "High-frequency common mode, the contaminator of signals." *Proceedings of the 39th International Instrumentation Symposium*, paper 93-070.

[7] Wayne, Scott, "Finding the Needle in a Haystack—Measuring small differential voltages in the presence of large common-mode voltages." *Analog Dialogue*, Volume 34 (2000), pp. 21-24. http://www.analog.com/library/analogDialogue/archives/34-01/haystack/index.html

[8] O'Grady, Albert, "Transducer/Sensor excitation and measurement techniques." *Analog Dialogue* Volume 34 (2000), pp. 33-37. http://www.analog.com/library/analogDialogue/archives/34-05/sensor/index.html

[9] www.scanivalve.com

PRODUCT INTRODUCTIONS: VOLUME 37, NUMBER 3

July

Power Output Stage for Class-D Audio Amplifier **AD1991**
High-performance **OOK/ASK/FSK/GFSK** Transmitter for ISM band systems **ADF7011**
Industry's first quad 256-position **Digital Potentiometer** with pin-selectable SPI/I²C interface **AD5263**

August

18-bit, 100-/570-/800-ksps **Successive-Approximation ADCs** accept fully differential inputs **AD7678/79/74**
4-GHz **Fractional-N Frequency Synthesizer** **ADF4153**
Dual 16-/14-bit serial-input, **Current-Output DACs**. **AD5545/55**
Logarithmic Converter provides 60-dB measurement range in optical communication systems **ADL5306**
1-bit, 2-port **Bus Switch** provides level translation between 1.8-V, 2.5-V, and 3.3-V systems **ADG3241**
Microprocessor Supervisory Circuits **ADM6711, ADM6713, ADM803**
10-bit 20-MHz **CCD Signal Processor** with Precision Timing™ Generator **AD9898**
10-bit 27-MHz **CCD Signal Processor** with Precision Timing Generator **AD9991**
Integrated Vcom and **Gamma Buffers** for flat panel LCD display applications **ADD8702**
Charge-Pump Regulator for color TFT LCD panels **ADM8832**
Multiformat **Video Encoder** has six 12-bit Noise Shaped Video (NSV™) DACs **ADV7310/11**

September

16-bit, 1-Msps **Successive-Approximation ADC** includes on-chip voltage reference. **AD7653**
16-/24-bit **Sigma-Delta ADCs** consume less than 75 microamps **AD7788/89**
16-/24-bit **Sigma-Delta ADCs** include differential input buffer **AD7790/91**
Low-distortion, high-speed dual **Op Amp** has rail-to-rail inputs and outputs. **AD8028**
Micropower precision **Op Amp** has rail-to-rail inputs and outputs. **AD8603**
Precision **Op Amp** combines low noise, low bias current, and low power **AD8671**
Broadband **Quadrature Demodulator** operates from 50 MHz to 1 GHz **AD8348**
IF Digitizing Subsystem operates from 10 MHz to 300 MHz. **AD9864**
32-bit floating-point **SHARC DSP Microcomputer** **ADSP-21160N**
Single-phase, multifunction **Energy-Metering IC** with di/dt sensor interface **ADE7753**
Precision, low-noise, 1.000-/1.200-V shunt **Voltage References** **ADR510/12**
Dual SPDT **CMOS Switches/2:1 Multiplexers** have 0.5-ohm *on*-resistance. **ADG836**
Wideband SPST **CMOS Switches** have 37-dB isolation at 1 GHz **ADG901/02**
Wideband SPDT **CMOS Switches/2:1 Multiplexers** have 37-dB isolation at 1 GHz. **ADG918/19**
Digital Temperature Sensor with quad voltage-output DAC is accurate to ±0.5°C **ADT7316/17/18**
10-bit **Digital Temperature Sensor** includes 8-channel ADC. **ADT7411**
Digital Temperature Sensor with 4-channel ADC and quad DAC is accurate to ±0.5°C. **ADT7516/17/18**
Complete 12-bit, 25-/40-MHz **CCD Signal Processors** **AD9944/45**
10-/12-bit, 25-/36-MHz **CCD Signal Processor** with Precision Timing Generator **AD9948/49**
12-bit 36-MHz **CCD Signal Processor** with Precision Timing Generator for 3-field area arrays. **AD9995**
Multiformat 216-MHz **Video Encoder** has six 14-bit Noise Shaped Video® (NSV) DACs. **ADV7314**
Multiformat **Video Encoder** has three 11-bit DACs. . . **ADV7330**

AUTHORS

Mercedes Casamayor (page 3) is an Applications Engineer in the analog-to-digital (A/D) converter group at Analog Devices, Limerick, Ireland. She was graduated in Industrial Engineering, specializing in Electronics and Automatics, from the *Escuela Técnica Superior de Ingenieros Industriales* (ETSII.) at Polytechnic University of Valencia (UPV), Spain. On completion of her degree she spent five months in the Centre for Biomedical Engineering at the University of Limerick. She joined Analog Devices in 2001. In her spare time she enjoys reading, swimming, and being with her friends.

Claire Croke (page 3) is an Applications Engineer at Analog Devices in Limerick, Ireland, working in the Precision Converter group. She currently has responsibility for analog-to-digital (A/D) conversion products and has also worked with the switch group. Claire holds a BEng from the University of Limerick. She joined Analog Devices in 1999 after completing her degree. In her spare time Claire enjoys reading, tennis, and walking.

Mark Looney (page 5) is a Senior Design Engineer at Analog Devices in Greensboro, NC. He joined ADI in 1998 and has since designed military-qualified dc-to-dc converters and high-speed A/D converter modules. He earned BS (1994) and MS (1995) degrees in Electrical Engineering from the University of Nevada, Reno, and has published several articles. Prior to joining ADI, he helped start IMATS, a vehicle electronics and traffic solution company, and designed dc-to-dc converters and EMI filters for space-level, radiation-hostile environments.

Lidwine Martinot (page 10) was graduated in 1997 with an Engineering Diploma from Telecom Paris and received an MSc in Signal Processing and Communication Systems from the University of Bristol, UK. She joined the ADI Wireless-DSP team two years ago, after four years working at Motorola. Over the last five months, most of Lidwine's leisure time was occupied with taking care of her baby girl.

Analog Dialogue

www.analog.com/analogdialogue **dialogue.editor@analog.com**

Analog Dialogue is the free technical magazine of Analog Devices, Inc., published continuously for 37 years—starting in 1967. It discusses products, applications, technology, and techniques for analog, digital, and mixed-signal processing. It is currently published in two editions—*online*, monthly at the above URL, and quarterly *in print*, as periodic retrospective collections of articles that have appeared online. In addition to technical articles, the online edition has timely announcements, linking to data sheets of newly released and pre-release products, and "Potpourri"—a universe of links to important and rapidly proliferating sources of relevant information and activity on the Analog Devices website and elsewhere. The *Analog Dialogue* site is, in effect, a "high-pass-filtered" point of entry to the **www.analog.com** site—the virtual world of *Analog Devices*. In addition to all its current information, the *Analog Dialogue* site has archives with all recent editions, starting from Volume 29, Number 2 (1995), plus three special anniversary issues, containing useful articles extracted from earlier editions, going all the way back to Volume 1, Number 1.

If you wish to subscribe to—or receive copies of—the print edition, please go to **www.analog.com/analogdialogue** and click on **<subscribe>**. Your comments are always welcome; please send messages to **dialogue.editor@analog.com** or to these individuals: *Dan Sheingold*, Editor **[dan.sheingold@analog.com]** or *Scott Wayne*, Managing Editor and Publisher **[scott.wayne@analog.com]**.

Volume 37, Number 3, 2003

Analog Dialogue

Worldwide Headquarters
One Technology Way
P.O. Box 9106
Norwood, MA
02062-9106 U.S.A.
Tel: 781.329.4700,
(1.800.262.5643,
U.S.A. only)
Fax: 781.326.8703

Analog Devices, Inc. Europe
c/o Analog Devices SA
17–19, rue Georges Besse
Parc de Haute
Technologie d'Antony
F-92182
Antony Cedex, France
Tel: 33.1.46.74.45.00
Fax: 33.1.46.74.45.01

Analog Devices, Inc. Japan Headquarters
New Pier Takeshiba
South Tower Building
1-16-1 Kaigan,
Minato-ku, Tokyo
105-6891, Japan
Tel: 813.5402.8210
Fax: 813.5402.1063

Analog Devices, Inc. Southeast Asia Headquarters
RBS Tower, Rm 4501-3
Times Square
1 Matheson Street
Causeway Bay
Hong Kong, PRC
Tel: 852.2.506.9336
Fax: 852.2.506.4755

M02000373-77-11/03