Eric Jacobsen and Richard Lyons

The Sliding DFT

The standard method for spectrum analysis in digital signal processing (DSP) is the discrete Fourier transform (DFT), typically implemented using a fast Fourier transform (FFT) algorithm. However, there are applications that require spectrum analysis only over a subset of the N center frequencies of an N-point DFT. A popular, as well as efficient, technique for computing sparse DFT results is the Goertzel algorithm that computes a single complex DFT spectral bin value for every N input time samples. This article describes a *sliding DFT* process whose spectral bin output rate is equal to the input data rate, on a sample-by-sample basis, with the advantage that it requires fewer computations than the Goertzel algorithm for real-time spectral analysis. In applications where a new DFT output spectrum is desired every sample, or every few samples, the sliding DFT is computationally simpler than the traditional radix-2 FFT. We'll start our sliding DFT discussion by providing a brief review of the Goertzel algorithm and use its behavior as a yardstick to evaluate the performance of the sliding DFT technique. Following that, we will examine stability issues regarding the sliding DFT implementation as well as review the process of frequency-domain convolution to accomplish time-domain windowing. Finally, a modified sliding DFT structure is proposed that provides improved computational efficiency.

Goertzel Algorithm

The Goertzel algorithm, used in dual-tone multifrequency decoding and phase-shift keying/frequency-shift keying modem implementations, is commonly used to compute DFT spectra [1]-[4]. The algorithm is implemented in the form of a second-order infinite impulse response (IIR) filter as shown in Figure 1. This filter computes a single DFT output (the kth bin of an N-point DFT) defined by

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi nk/N}. \tag{1}$$

The filter's $y(n)$ output is equal to the DFT output frequency coefficient, $X(k)$, at the time index $n = N$. For emphasis, we remind the reader that the filter's $y(n)$ output is not equal to $X(k)$ at any time index when $n \neq N$. The frequency-domain index k is an integer in the range $0 \leq k \leq N-1$. The derivation of this filter's structure is readily available in the literature [5]-[7].

The z-domain transfer function of the Goertzel filter is

$$H_G(z) = \frac{1 - e^{-j2\pi k/N} z^{-1}}{1 - 2\cos(2\pi k / N) z^{-1} + z^{-2}} \tag{2}$$

with a single z-domain zero located at $z = e^{-j2\pi k/N}$ and conjugate poles at $z = e^{\pm j2\pi k/N}$ as shown in Figure 2(a). The pole/zero pair at $z = e^{-j2\pi k/N}$ cancels each other. The frequency magnitude response, provided in Figure 2(b), shows resonance centered at a normalized frequency of $2\pi k/N$, corresponding to a cyclic frequency $k \cdot f_s / N$ Hz (where f_s is the signal sample rate).

While the Goertzel algorithm is derived from the standard DFT equation, it's important to realize that the filter's frequency magnitude response is not the $\sin(x)/(x)$-like response of a single-bin DFT. The Goertzel filter is a complex resonator having an infinite-length unit impulse response, $h(n) = e^{j2\pi nk/N}$, and that's why its magnitude response is so narrow. The time-domain difference equations for the Goertzel filter are

$$v(n) = 2\cos(2\pi k/N) v(n-1) - v(n-2) + x(n) \tag{3a}$$

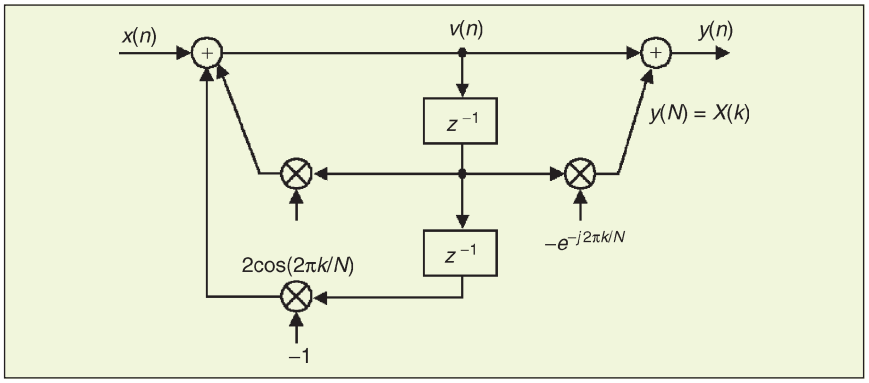
$$y(n) = v(n) - e^{-j2\pi k/N} v(n-1). \tag{3b}$$

An advantage of the Goertzel filter in calculating an N-point $X(k)$ DFT

"DSP Tips and Tricks" introduces practical tips and tricks of design and implementation of signal processing algorithms so that you may be able to incorporate them into your designs. We welcome readers who enjoy reading this column to submit their contributions. Please contact Associate Editor Rick Lyons at ricklyon@onemain.com.

1053-5888/03/$17.00©2003IEEE

bin is that (3a) is implemented N times while (3b), the feed forward path in Figure 1, need only be computed once after the arrival of the Nth input sample. Thus for real $x(n)$ the filter requires $N+2$ real multiplies and $2N+1$ real adds to compute an N-point $X(k)$. However, when modeling the Goertzel filter if the time index begins at $n=0$, the filter must process $N+1$ time samples with $x(N)=0$ to compute $X(k)$. Now let's look at the sliding DFT process.



▲ *1. IIR filter implementation of the Goertzel algorithm.*

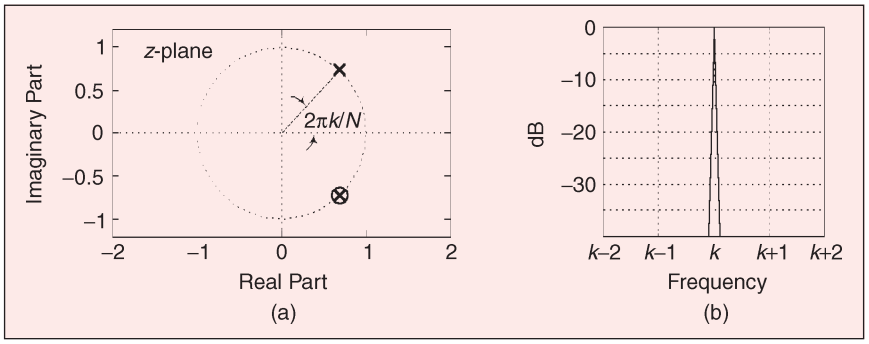
Sliding DFT

The sliding DFT (SDFT) algorithm performs an N-point DFT on time samples within a sliding-window as shown in Figure 3. In this example the SDFT initially computes the DFT of the $N=16$ time samples in Figure 3(a). The time window is then advanced one sample, as in Figure 3(b), and a new N-point DFT is calculated. The value of this process is that each new DFT is efficiently computed directly from the results of the previous DFT. The incremental advance of the time window for each output computation is what leads to the name *sliding DFT* or *sliding-window DFT*.

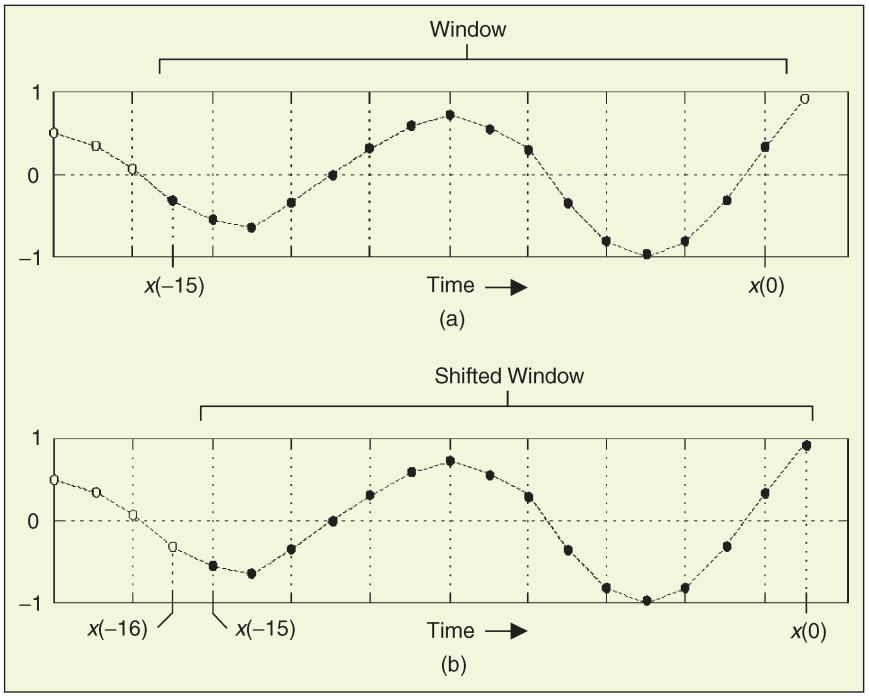
The principle used for the SDFT is known as the DFT shifting theorem or the circular shift property [8]. It states that if the DFT of a windowed (finite-length) time-domain sequence is $X(k)$, then the DFT of that sequence, circularly shifted by one sample, is $X(k)e^{j2\pi k/N}$. Thus the spectral components of a shifted time sequence are the original (unshifted) spectral components multiplied by $e^{j2\pi k/N}$, where k is the DFT bin of interest. We express this process by

$$S_k(n) = S_k(n-1)e^{j2\pi k/N} - x(n-N) + x(n) \qquad (4)$$

where $S_k(n)$ is the new spectral component and $S_k(n-1)$ is the previous spectral component. The subscript k



▲ *2. Goertzel filter: (a) z-domain pole/zero locations and (b) frequency magnitude response.*



▲ *3. Signal windowing for two 16-point DFTs: (a) data samples in the first computation and (b) second computation samples.*

reminds us that the spectra are those associated with the kth DFT bin.

Equation (4), whose derivation is provided in the Appendix, reveals the value of this process in computing real-time spectra. We calculate $S_k(n)$ by phase shifting the previous $S_k(n-1)$ components, subtract the $x(n-N)$ sample, and add the current $x(n)$ sample. Thus the SDFT requires only one complex multiply and two real adds per output sample. The computational complexity of each successive N-point output is then $O(N)$ for the sliding DFT compared to $O(N^2)$ for the DFT and $O[N \log_2(N)]$ for the FFT. Unlike the DFT or FFT, however, due to its recursive nature the sliding DFT output *must* be computed for each new input sample. If a new N-point DFT output is required only every N inputs, the sliding DFT requires $O(N^2)$ computations and is equivalent to the DFT. When output computations are required every M input samples, and M is less than $\log_2(N)$, the sliding DFT can be computationally superior to traditional FFT implementations even when all N DFT outputs are required.

Equation (4) leads to the single-bin SDFT filter structure shown in Figure 4.

The single-bin SDFT algorithm is implemented as an IIR filter with a comb filter followed by a complex resonator [9]. (If you want to compute all N DFT spectral components, N resonators with $k = 0$ to $N-1$ will be needed, all driven by a single comb filter.) The comb filter delay of N samples forces the filter's transient response to be $N-1$ samples in length, so the output will not reach steady state until the $S_k(N)$ sample. In practical applications the algorithm can be initialized with zero input and zero output. The output will not be valid, or equivalent to (1)'s $X(k)$, until N input samples have been processed. The z-domain transfer function for the kth bin of the sliding DFT filter is

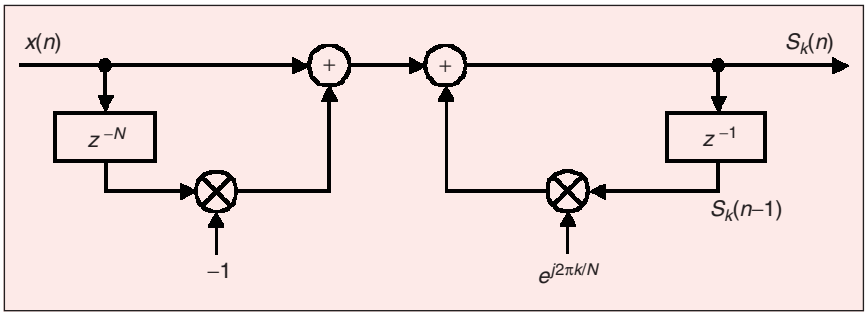
$$H_{SDFT}(z) = \frac{(1 - z^{-N})}{1 - e^{j2\pi k/N} z^{-1}}. \quad (5)$$

This complex filter has N zeros equally spaced around the z-domain's unit circle, due to the N-delay comb filter, as well as a single pole canceling the zero at $z = e^{j2\pi k/N}$. The SDFT filter's complex unit impulse response $h(n)$ and pole/zero locations are shown in Figure 5 for the example where $k = 2$ and $N = 20$.

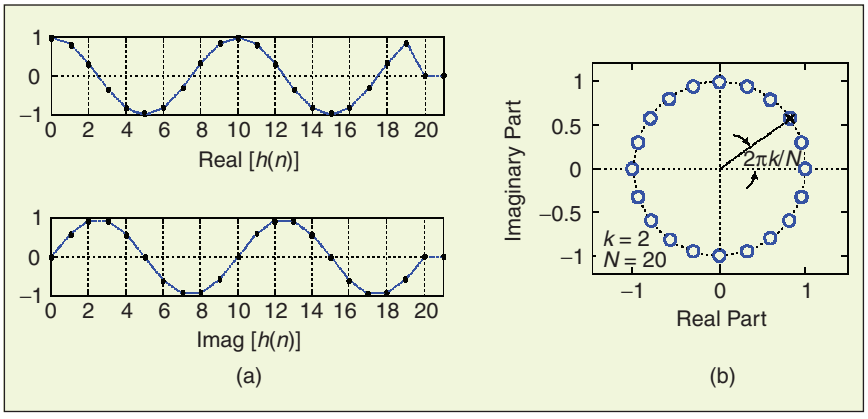
Because of the comb subfilter, the SDFT filter's complex sinusoidal unit impulse response is finite in length, truncated in time to N samples, and that property makes the frequency magnitude response of the SDFT filter identical to the $\sin(Nx)/\sin(x)$ response of a single DFT bin centered at a normalized frequency of $2\pi k/N$.

One of the attributes of the SDFT is that once an $S_k(n-1)$ is obtained, the number of computations to calculate $S_k(n)$ is fixed and independent of N. A computational workload comparison between the Goertzel and SDFT filters is provided later in this article. Unlike the radix-2 FFT, the SDFT's N can be any positive integer giving us greater flexibility to tune the SDFT's center frequency by defining integer k such that $k = N \cdot f_i / f_s$, when f_i is a frequency of interest in hertz. In addition, the SDFT does not require bit-reversal processing as does the FFT. Like Goertzel, the SDFT is especially efficient for narrowband spectrum analysis.

For completeness, we mention that a radix-2 sliding FFT technique exists for computing all N bins of $X(k)$ in (1) [10], [11]. This method is computationally attractive because it requires only N complex multiplies to update the N-point FFT for all N bins; however, it requires $3N$ memory locations ($2N$ for data and N for twiddle coefficients). Unlike the



▲ 4. Single-bin sliding DFT filter structure.



▲ 5. Sliding DFT characteristics for $k = 2$ and $N = 20$: (a) impulse response and (b) pole/zero locations.

SDFT, the radix-2 sliding FFT scheme requires address bit-reversal processing and restricts N to be an integer power of two.

SDFT Stability

The SDFT filter is only marginally stable because its pole resides on the z-domain's unit circle. If filter coefficient numerical rounding error is not severe, the SDFT is bounded-input, bounded-output stable. Filter instability can be a problem, however, if numerical coefficient rounding causes the filter's pole to move outside the unit circle. We can use a damping factor r to force the pole to be at a radius of r inside the unit circle and guarantee stability using a transfer function of

$$H_{\text{SDFT},gs}(z) = \frac{(1 - r^N z^{-N})}{1 - re^{j2\pi k/N} z^{-1}} \tag{6}$$

with the subscript gs meaning guaranteed-stable. The stabilized feed-forward and feedback coefficients become $-r^N$ and $re^{j2\pi k/N}$, respectively. The difference equation for the stable SDFT filter becomes

$$S_{k,gs}(n) = S_{k,gs}(n-1)re^{j2\pi k/N} - x(n-N)r^N + x(n) \tag{7}$$

with the stabilized-filter structure shown in Figure 6.

Using a damping factor as in Figure 6 guarantees stability, but the $S_k(n)$ output, defined by

$$X_{r<1}(k) = \sum_{n=0}^{N-1} x(n) \cdot re^{-j2\pi nk/N} \tag{8}$$

is no longer exactly equal to the kth bin of an N-point DFT in (1). While the error is reduced by making r very close to (but less than) unity, a scheme does exist for eliminating that error completely once every N output samples at the expense of additional conditional logic operations [12]. Determining if the damping factor r is necessary for a particular SDFT application requires careful empirical investigation.

Another stabilization method worth consideration is decrementing the largest component (either real or imaginary) of the filter's $e^{j2\pi k/N}$ feedback coefficient by one least significant bit. This technique can be applied selectively to problematic output bins and is effective in combating instability due to rounding errors which result in finite-precision $e^{j2\pi k/N}$ coefficients having magnitudes greater than unity.

Like the DFT, the SDFT's output is proportional to N, so in fixed-point binary implementations the designer must allocate sufficiently wide registers to hold the computed results.

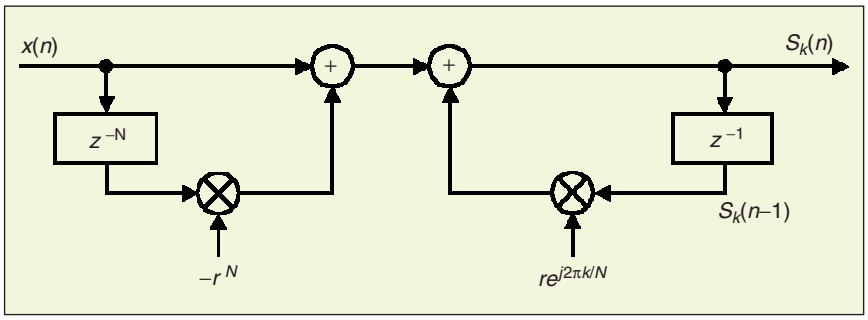
Time-Domain Windowing in the Frequency Domain

The spectral leakage of the SDFT can be reduced by the standard concept of windowing the $x(n)$ input time samples. However, windowing by time-domain multiplication would compromise the computational simplicity of the SDFT. Alternatively, we can implement a time-domain window by means of frequency-domain convolution.

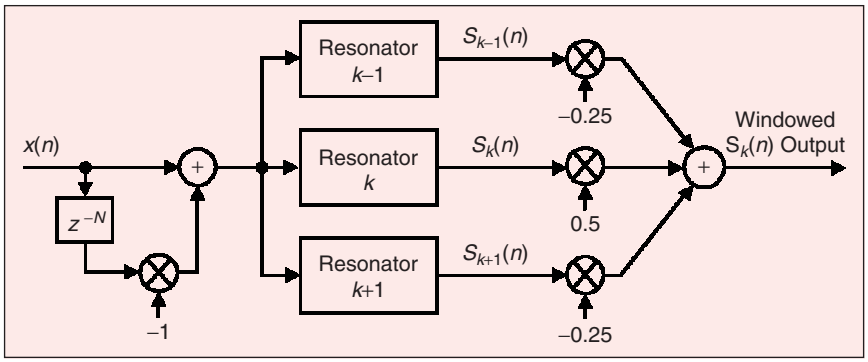
Spectral leakage reduction performed in the frequency domain is accomplished by convolving adjacent $S_k(n)$ values with the DFT of a window function. For example, the DFT of a Hanning window comprises only three nonzero values, –0.25, 0.5, and –0.25. As such we can compute a Hanning-windowed $S_k(n)$, the kth DFT bin, with a three-point convolution using

$$\text{Hanning-windowed } S_k(n) = -0.25 \cdot S_{k-1}(n) + 0.5 \cdot S_k(n) - 0.25 \cdot S_{k+1}(n). \tag{9}$$

Figure 7 shows this process where the comb filter stage need only be implemented once. Thus a Hanning window can be implemented by binary right shifts (assuming integer



▲ 6. Guaranteed-stable sliding DFT filter structure.



▲ 7. Three-resonator structure to compute three SDFT bin results and a three-point convolution.

arithmetic) and two complex adds for each SDFT bin, making the Hanning window attractive in ASIC and FPGA implementations where single-cycle hardware multiplies are costly. If a gain of four is acceptable, then only one left shift two complex adds are required using

Hanning-windowed
$$S_k(n) = -S_{k-1}(n) + 2 \cdot S_k(n) - S_{k+1}(n). \quad (10)$$

The Hanning window is a member of a category called $\cos^{\alpha}(x)$ window functions [13], [14]. These functions are also known as generalized cosine windows because their N-point time-domain samples are defined as

$$w(n) = \sum_{m=0}^{\alpha-1} (-1)^m a_m \cos(2\pi mn/N) \quad (11)$$

where $n = 0,1,2,\ldots,N-1$, and the integer α specifies the number of terms in the window's time function. These window functions are attractive for frequency domain convolution because their DFTs contain only a few nonzero samples. The frequency domain vectors of various $\cos^{\alpha}(x)$ window functions follow the form $(1/2)\cdot(a_2, -a_1, 2a_0, -a_1, a_2)$, with a few examples presented in Table 1. Additional $\cos^{\alpha}(x)$ window functions are described in the literature [14].

Sliding Goertzel DFT

We can reduce the number of multiplications required in the SDFT by creating a new pole/zero pair in its $H_{DFT}(z)$ system function [7]. This is done by multiplying the numerator and denominator of $H_{SDFT}(z)$ in (5) by the factor $(1 - e^{-j2\pi k/N} z^{-1})$ yielding

$$H_{SG}(z) = \frac{(1 - e^{-j2\pi k/N} z^{-1})(1 - z^{-N})}{(1 - e^{-j2\pi k/N} z^{-1})(1 - e^{j2\pi k/N} z^{-1})} = \frac{(1 - e^{-j2\pi k/N} z^{-1})(1 - z^{-N})}{1 - 2\cos(2\pi k/N) z^{-1} + z^{-2}} \quad (12)$$

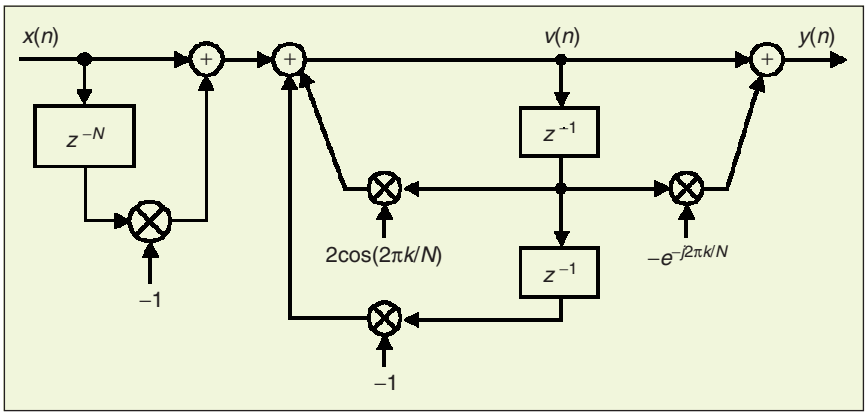
where the subscript SG means sliding Goertzel. The filter block diagram for $H_{SG}(z)$ is shown in Figure 8 where this new filter is recognized as the standard Goertzel filter preceded by a comb filter. The sliding Goertzel DFT filter, unlike the standard Goertzel filter, has a finite-duration impulse response identical to that shown in Figure 5(a), for $k = 2$ and $N = 20$.

Of course, unlike the traditional Goertzel filter in Figure 1, the sliding Goertzel DFT filter's complex feedforward computations must be performed for each input time sample. The sliding Goertzel filter's $\sin(Nx)/\sin(x)$ frequency magnitude response, for $k = 2$ and $N = 20$, is provided in Figure 9(a). The asymmetrical frequency response is defined by the filter's N zeros equally spaced around the z-domain's unit circle in Figure 9(b) due to the N-delay comb filter, as well as an additional (uncanceled) zero located at $z = e^{-j2\pi k/N}$ on account of the $(1 - e^{-j2\pi k/N} z^{-1})$ factor in the $H_{SG}(z)$ transfer function's numerator. In addition, the filter has conjugate poles canceling zeros at $z = e^{\pm j2\pi k/N}$.

The sliding Goertzel DFT filter is of interest because its computational workload is less than that of the SDFT. This is because the $v(n)$ samples in Figure 8 are real-only due to the real-only feedback coefficients. A single-bin DFT computational comparison, for real-only inputs, is provided in Table 2. For real-time processing requiring spectral updates on a sample by sample basis the slid-

Table 1. cosα(x) windows, frequency domain coefficients.

Window function	α_0	α_1	α_2
Rectangular	1.0	–	–
Hanning ($\alpha = 2$)	0.5	0.25	–
Hamming ($\alpha = 2$)	0.54	0.46	–
Blackman ($\alpha = 3$)	0.42	0.5	0.08
Exact Blackman ($\alpha = 3$)	7938/18608	9240/18608	1430/18608



▲ 8. Structure of the sliding Goertzel DFT filter.

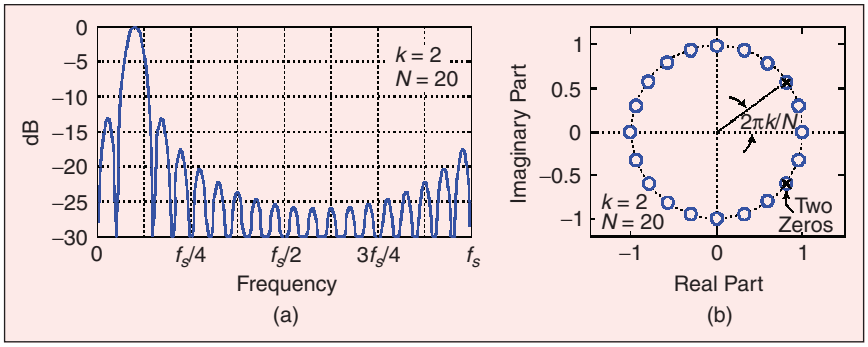
ing Goertzel method requires fewer multiplies than either the SDFT or the traditional Goertzel algorithm.

Summary

The sliding DFT process for spectrum analysis was presented and shown to be more efficient than the popular Goertzel algorithm for sample-by-sample DFT bin computations. The sliding DFT provides computational advantages over the traditional DFT or FFT for many applications requiring successive output calculations, especially when only a subset of the DFT output bins are required. Methods for output stabilization as well as time-domain data windowing by means of frequency- domain convolution were also discussed. A modified sliding DFT algorithm, called the sliding Goertzel DFT, was proposed to further reduce computational workload.

Eric Jacobsen is minister of algorithms at Intel. He currently leads the Advanced OFDM wireless research effort within Intel Labs and has interests in channel modeling, efficient algorithms, synchronization, coding, adaptive techniques, and other aspects of wireless communication systems. He has developed efficient hardware and software implementation techniques for signal processing in radar, imaging, satellite, and communications systems. With an M.S.E.E. from South Dakota School of Mines and Technology, he is a member of the IEEE and the Eta Kappa Nu honor society and occasionally road races a 1995 Taurus SHO.

Richard Lyons is a consulting systems engineer and lecturer with Besser Associates in Mt. View, California. He has been the lead hardware engineer for numerous multimillion dollar signal processing systems for both the National Security Agency (NSA) and TRW Inc. and has taught at the University of California Santa Cruz Extension. He is an associate editor for *IEEE Signal Processing Magazine* and author of *Understanding Digital Signal Processing* (Prentice-Hall, 1997). He is a member of the IEEE and the Eta Kappa Nu honor society and rides a 1981 Harley Davidson.



▲ *9. Sliding Goertzel filter for N = 20 and k = 2: (a) frequency magnitude response and (b) z-domain pole/zero locations.*

Table 2. Single-bin DFT comparison.

	Single $S_k(n)$ Computation		Next $S_k(n+1)$ Computation	
	Real Multiplies	Real Adds	Real Multiplies	Real Adds
DFT	$2N$	$2N$	$2N$	$2N$
Standard Goertzel	$N + 2$	$2N + 1$	$N + 2$	$2N + 1$
Sliding DFT	$4N$	$4N$	4	4
Sliding Goertzel DFT	$N + 2$	$3N + 1$	3	4

References

[1] M. Felder, J. Mason, and B. Evans, "Efficient dual-tone multi-frequency detection using the non-uniform discrete Fourier transform," *IEEE Signal Processing Lett.*, vol. 5, pp. 160-163, July 1998.

[2] *Using the ADSP-2100 Family*, vol. 1. Norwood, MA: Analog Devices, 1995, chap. 14 [Online]. Available: http://www.analog.com/Analog_Root/static/library/dspManuals/Using_ADSP-2100_Vol1_books.html

[3] S. Gay, J. Hartung, and G. Smith, "Algorithms for multi-channel DTMF detection for the WERDSP32 family," in *Proc. Int. Conf. ASSP*, 1989, pp. 1134-1137.

[4] K. Banks, "The Goertzel algorithm," *Embedded Syst. Programming Mag.*, pp. 34-42, Sept. 2002.

[5] G. Goertzel, "An algorithm for the evaluation of finite trigonometric series," *American Math. Month.*, vol. 65, pp. 34-35, 1958.

[6] J. Proakis and D. Manolakis, *Digital Signal Processing-Principles, Algorithms, and Applications*, 3rd ed. Upper Saddle River, NJ: Prentice Hall, 1996, pp. 480-481.

[7] A. Oppenheim, R. Schafer, and J. Buck, *Discrete-Time Signal Processing*, 2nd ed. Upper Saddle River, NJ: Prentice Hall, 1996, pp. 633-634.

[8] T. Springer, "Sliding FFT computes frequency spectra in real time," *EDN Mag.*, pp. 161-170, Sept. 29, 1988.

[9] L. Rabiner and B. Gold, *Theory and Application of Digital Signal Processing*. Upper Saddle River, NJ: Prentice Hall, 1975, pp. 382-383.

[10] B. Farhang-Boroujeny and Y. Lim, "A comment on the computational complexity of sliding FFT," *IEEE Trans. Circuits Syst. II*, vol. 39, no. 12, pp. 875-876, Dec. 1992.

[11] B. Farhang-Boroujeny and S. Gazor, "Generalized sliding FFT and its application to implementation of block LMS adaptive filters," *IEEE Trans. Signal Processing*, vol. 42, no. 3, pp. 532-538, Mar. 1994.

[12] S. Douglas and J. Soh, "A numerically stable sliding-window estimator and its application to adaptive filters," in *Proc. 31st Annual Asilomar Conf. on Signals, Systems, and Computers,* Pacific Grove, CA, vol. 1, Nov. 1997, pp. 111-115.

[13] F. Harris, "On the use of windows for harmonic analysis with the discrete Fourier transform," *Proc. IEEE*, vol. 66, pp. 51-84, Jan. 1978.

[14] A. Nuttall, "Some windows with very good sidelobe behavior," *IEEE Trans. Acoust., Speech, Signal Processing,*, vol. 29, pp. 84-91, Feb. 1981.

Appendix

The derivation of the SDFT time-domain expression is straightforward by example. We start by using the principles of the z-transform and write a generalized z-domain spectrum of a four-point time-domain sequence $x(n)$, where $n = 0,1,\ldots,3$, evaluated at $z = z_o$ as

$$S_{z_o}(0) = x(3) + x(2)z_o + x(1)z_o^2 + x(0)z_o^3. \quad \text{(A1)}$$

Likewise we could compute a spectrum of the four $x(n+1)$ time samples as

$$S_{z_o}(1) = x(4) + x(3)z_o + x(2)z_o^2 + x(1)z_o^3. \quad \text{(A2)}$$

If we multiply $S_{z_o}(0)$ by z_o we have

$$S_{z_o}(0)z_o = x(3)z_o + x(2)z_o^2 + x(1)z_o^3 + x(0)z_o^4. \quad \text{(A3)}$$

Comparing (A2) and (A3) we can rewrite (A2) as

$$S_{z_o}(1) = S_{z_o}(0)z_o - x(0)z_o^4 + x(4). \quad \text{(A4)}$$

Thus we can use $S_{z_o}(0)$ to compute $S_{z_o}(1)$. Moreover we can express the spectrum of the four $x(n+2)$ samples as

$$S_{z_o}(2) = S_{z_o}(1)z_o - x(1)z_o^4 + x(5). \quad \text{(A5)}$$

In the general case, the qth N-point spectrum can be expressed as

$$S_{z_o}(q) = S_{z_o}(q-1)z_o - x(q-1)z_o^N + x(q+N-1). \quad \text{(A6)}$$

Here's the payoff for our efforts. If we let $z_o = e^{j2\pi k/N}$, a point on the unit circle, the (A6) z-transform expression becomes the desired time-domain expression for the sliding DFT as

$$\begin{aligned} S_{e^{j2\pi k/N}}(q) &= S_{e^{j2\pi k/N}}(q-1)e^{j2\pi k/N} \\ &\quad - x(q-1)e^{j2\pi kN/N} \\ &\quad + x(q+N-1) \\ &= S_{e^{j2\pi k/N}}(q-1)e^{j2\pi k/N} \\ &\quad - x(q-1) + x(q+N-1). \end{aligned} \quad \text{(A7)}$$

Because the angle of $e^{j2\pi k/N}$ is an integer multiple of $2\pi/N$, (A7) is seen merely as the qth single-bin DFT, $X(k)$, of $x(n)$ for the normalized frequency of $2\pi k/N$ radians, where $k = 0,1,2,\ldots,N-1$. We now have an expression for the sliding DFT. However, to turn that expression into a filter difference equation we're compelled to modify the indices of (A7) to make them compatible with causal filters. With no loss in generality, using the following substitutions

$$\begin{aligned} S_k(n) &= S_{e^{j2\pi k/N}}(q), \\ S_k(n-1) &= S_{e^{j2\pi k/N}}(q-1), \\ x(n) &= x(q+N-1), \\ x(n-N) &= x(q-1), \end{aligned} \quad \text{(A8)}$$

we can rewrite (A7) in a time-domain form for the causal recursive filter

$$S_k(n) = S_k(n-1)e^{j2\pi k/N} - x(n-N) + x(n). \quad \text{(A9)}$$

The subscript k reminds us that the filter output is associated with the kth DFT bin. The z-domain transfer function for the kth bin of the sliding DFT filter is

$$H_{\text{SDFT}}(z) = \frac{(1-z^{-N})}{1-e^{j2\pi k/N}z^{-1}}. \quad \text{(A10)}$$