Fred Harris

Ultra Low Phase Noise DSP Oscillator

Many DSP algorithms in analysis and communication systems require a complex sinusoid to accomplish various signal rotation tasks. Examples include the discrete and fast Fourier transforms, digital up/down conversions, and operations in communication carrier recovery loops [1]. Often the desired sine and cosine samples of the complex sinusoid are generated by a computationally efficient oscillator, called a direct digital synthesizer (DDS) and implemented with the CORDIC algorithm [2]. In a CORDIC DDS the phase noise (phase angle error) is inversely proportional to the number of CORDIC iterations performed. As such, to improve the accuracy (i.e., reduce the phase noise) of the sine and cosine samples, additional CORDIC iterations need to be performed.

This article describes a novel complex oscillator, which is based on an interesting variation of the traditional CORDIC DDS and produces sine and cosine output samples of any specified angle. Our oscillator provides drastically improved phase noise performance (relative to a traditional CORDIC DDS) without the need for additional CORDIC iteration processing. In addition, our oscillator supports real-time output sample-by-sample digital frequency control.

In describing our enhanced complex oscillator, we first present the arithmetic processing needed to generate sine and cosine samples. Next we show how that processing is implemented using the CORDIC algorithm. Then we detail the trick used to reduce oscillator phase noise errors. Finally we show how to guarantee a stable oscillator output amplitude and present an example of our oscillator's ultra low phase noise performance.

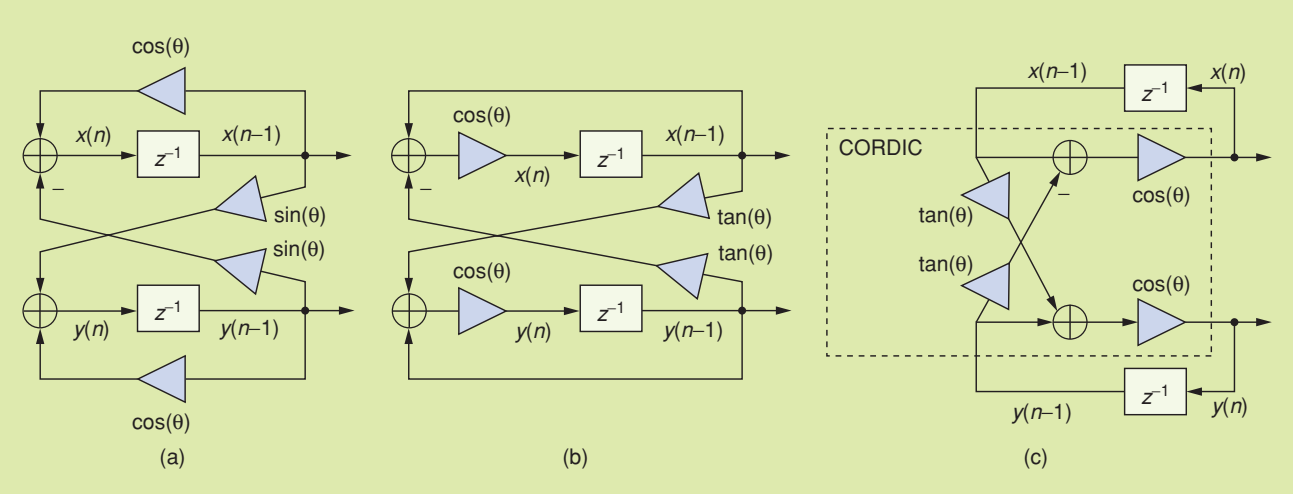
"DSP Tips and Tricks" introduces practical design and implementation signal processing algorithms that you may wish to incorporate into your designs. We welcome readers to submit their contributions to Associate Editors Rick Lyons (r.lyons@ieee.org) or Britt Rorabaugh (dspboss@aol.com).

OSCILLATOR OUTPUT SEQUENCE GENERATION

Consider a complex oscillator with output samples $w(n)$ given by

$$w(n) = \exp(j\theta) \cdot w(n-1), \text{ where } n = 0, 1, 2, \ldots \text{ and } w(-1) = 1 + j0. \quad (1)$$

An oscillator that implements (1) operates at a digital frequency of $\theta = 2\pi f_C/f_S$ radians/sample. Frequency θ represents the oscillator's change in angle per unit time, and frequencies f_C and f_S are the oscillator's output cyclic frequency and sample rate, respectively, in hertz. Using $w(n) = \cos(n) + j\sin(n)$ we can write



[FIG1] **Block diagrams of oscillator networks: (a) traditional form; (b) alternate form; (c) desired recursive CORDIC form.**

1053-5888/07/$25.00©2007IEEE

[TABLE 1] CORDIC θ_k ROTATION ANGLES FOR DIFFERENT k.

k	2^{-k}	$\theta_k = \text{atan}(2^{-k})$, (DEGREES)	$\cos(\theta_k) = \frac{1}{\sqrt{1+2^{-2k}}}$ (RADIANS)
0	1.0	45.0000000	0.70710678
1	0.5	26.5650510	0.89442719
2	0.25	14.0362430	0.97014250
3	0.125	7.1250160	0.99227787
4	0.0625	3.5763340	0.99805257
5	0.03125	1.7899110	0.99951207
6	0.015625	0.8951737	0.99987795
7	0.0078125	0.4476142	0.99996948
8	0.00390625	0.2238105	0.99999237
9	0.001953125	0.1119057	0.99999809

$$\begin{aligned} x(n) =& x(n-1)\cos(\theta) \\ & - y(n-1)\sin(\theta) : x(-1) = 1 \\ y(n) =& x(n-1)\cos(\theta) \\ & - y(n-1)\sin(\theta) : y(-1) = 0. \end{aligned} \tag{2}$$

Sequences $x(n)$ and $y(n)$ are the desired cosine and sine oscillator outputs, respectively. The remainder of this article describes how to accurately generate these two sequences.

The traditional structure of a complex oscillator that implements (2) is illustrated in the block diagram of Figure 1(a). Next, from this structure we can obtain an alternate form, which is illustrated in Figure 1(b), by factoring the cosine terms from (2) as

$$\begin{aligned} x(n) =& [x(n-1) \\ & - y(n-1)\tan(\theta)]\cos(\theta) \\ y(n) =& [y(n-1) \\ & + x(n-1)\tan(\theta)]\cos(\theta). \end{aligned} \tag{3}$$

Then, by rearranging the computations in Figure 1(b) we can draw our desired Figure 1(c) oscillator wherein the $\tan(\theta)$ multiplications can be efficiently implemented by the CORDIC algorithm, without the need for multipliers.

CORDIC ALGORITHM-BASED PROCESSING

CORDIC tan(Θ) COMPUTATION

To understand how processing is implemented using the CORDIC algorithm, consider the structure illustrated in Figure 1(c). If the digital frequency θ is a known fixed value, we can compute $\tan(\theta)$ and $\cos(\theta)$ offline. On the other hand, if the digital frequency θ is to be programmable or continuously variable, we have the problem of computing the $\tan(\theta)$ and $\cos(\theta)$ terms for an arbitrary angle θ. We respond to this task by replacing the $\tan(\theta)$ multiplications in (3) with a sequence of elementary rotations known as the CORDIC rotate. We convert a multiplication by $\tan(\theta)$ to trivial shift-and-add operations by selecting rotation angles θ_k that satisfy

$$\tan(\theta_k) = 2^{-k},\ k = 0, 1, 2, \ldots, K-1. \tag{4}$$

In practice the θ_k angles, which are explicitly listed in Table 1, are stored in a look-up table memory. Any angle θ in the first quadrant can be approximated by a binary (± 1) weighted sum of the angles, such as

$$\theta = \sum_{k=0}^{9} \alpha_k \theta_k + \theta_{\text{Rem}};\ \alpha_k = \pm 1. \tag{5}$$

where θ_{Rem} is a residual error. (Values of θ larger than 90 degrees are accommodated by multiplying with j.) For example, a ten-term approximation results in an angle error less than $\text{atan}(2^{-10})$, which is approximately equal to 0.056 degrees. Later we will see how this residual error θ_{Rem} is folded into the rotation as a final clean-up correction to greatly improve the performance of our DDS.

To successively approximate the desired $\tan(\theta)$ in K rotations (iterations) we use a state machine that cycles through the sequence of binary shifts and adds to approximate the desired product. An initial $-\theta$ value stored in the angle accumulator is driven by the CORDIC system toward zero by adding or subtracting the successive angles accessed from the arctangent look-up table in Figure 2.

CORDIC cos(Θ) COMPUTATION

As the CORDIC algorithm proceeds to compute $\tan(\theta)$, the $\cos(\theta)$ multiply operation in Figure 1(c) remains to be performed. To limit the number of arithmetic operations, instead of executing a $\cos(\theta_k)$ multiplication during each CORDIC rotation, we perform the $\cos(\theta)$ multiply only once at the end of the rotation sequence. For θ in the first quadrant, recalling that $\cos\theta(k) = 1/\text{sqrt}[1 + \text{atan}^2(\theta_k)]$, the $\cos(\theta)$ factor for $K = 10$ CORDIC rotations is given by

$$\begin{aligned} \cos(\theta) &= \prod_{k=0}^{9} \cos(\theta_k) \\ &= \prod_{k=0}^{9} \frac{1}{\sqrt{1+2^{-2k}}} \\ &= 0.596495. \end{aligned} \tag{6}$$

So, at the end of the CORDIC rotations, we multiply each CORDIC output by $\cos(\theta) = 0.596495$. (Note that the right-most column in Table 1 contains the individual $\cos(\theta_k)$ factors whose cumulative product is 0.596495.) As an added benefit, the $\cos(\theta)$ multiply exactly compensates for the undesired output amplitude increase inherent in the CORDIC algorithm.

LOW PHASE NOISE CORDIC OSCILLATOR

DESIGN

With these notes in mind, we now obtain the structure of our final recursive CORDIC oscillator as shown in Figure 2. In this oscillator, for each complex output sample the negative of the desired frequency angle θ is inserted in the angle accumulator. The CORDIC rotation engine performs (for instance) $K = 10$ iterations of binary shift-and-add operations of the ordered pairs $[x(n-1),$

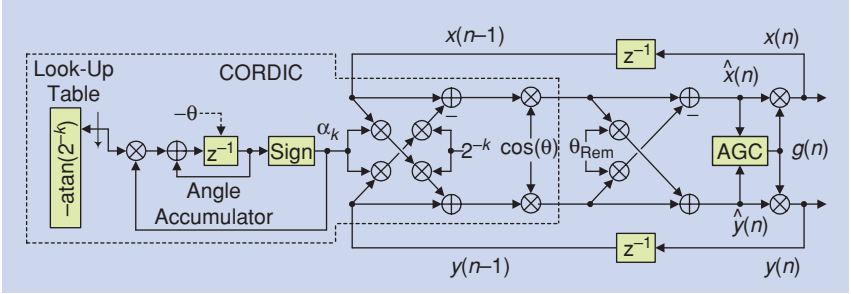
$y(n-1)]$ while trying to zero the content of the angle accumulator by adding or subtracting the angles $\theta_k = \text{atan}(2^{-k})$ stored in the arctangent look-up table. Based on the sign of the current-iteration angle accumulator content, the Sign function in Figure 2, whose output is ± 1, determines whether an angle addition or angle subtraction takes place. The $\cos(\theta)$ multiply is applied once, at the end (after the tenth rotation) rather than once per rotation. After the tenth CORDIC rotation there is assuredly a nonzero residual angle θ_{Rem} in the angle accumulator. So our DSP trick is that we perform a clean-up rotation by advancing the CORDIC's complex output angle by the correction angle θ_{Rem}. This phase angle correction, whose derivation is described in [3], is what suppresses first-order phase noise in the DDS.

OUTPUT STABILIZATION

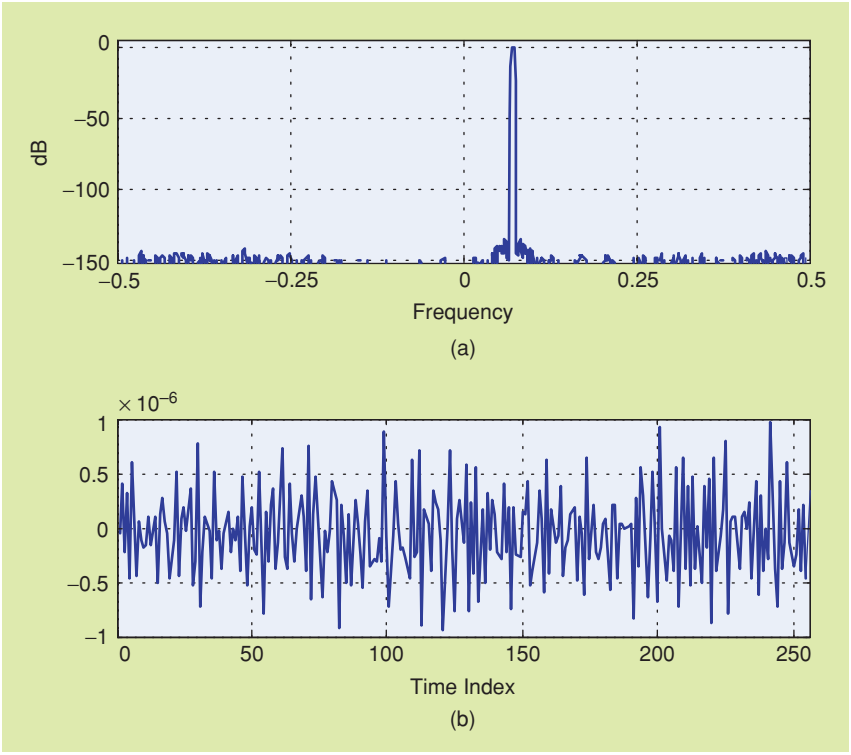
Upon analysis of the network in Figure 1(a) we find that its z-domain single-pole location resides on a Cartesian grid with the real part of the pole equal to $\cos(\theta)$ and the imaginary part of the pole equal to $\sin(\theta)$. Because the sine and cosine (or the tangent and cosine) in the network are transcendental numbers, and the arithmetic implementing the network has finite precision, the system pole can not lie precisely on the unit circle. Thus, the pole is either inside or outside the unit circle and the response of the network to an initial condition is an exponentially decaying or growing sinusoid. To use our DDS as a quadrature signal generator we must incorporate an automatic gain control (AGC) loop to stabilize its output amplitude as we want to keep the system pole on the unit circle. The AGC that we employ is shown in Figure 2, where the data-dependent $g(n)$ gain correction, needed to ensure the oscillator's output magnitude remain at unity, is given by

$$g(n) = \frac{3 - [\hat{x}^2(n) + \hat{y}^2(n)]}{2}. \qquad (7)$$

where $\hat{x}$ and $\hat{y}$ are the outputs of the θ_{Rem} rotation operation [3]–[5]. The $g(n)$ gain term in (7), applied to the output of the θ_{Rem} phase angle correction process,



[FIG2] **Recursive CORDIC DDS with automatic level control.**



[FIG3] **Complex CORDIC oscillator: (a) spectrum; (b) small angle and finite-arithmetic gain correction time series.**

operates in the direction to correct the amplitude error caused by the final phase correction, and the amplitude increases or decreases due to the pole position error relative to the unit circle.

EXAMPLE

Consider an example where the CORDIC algorithm ran ten iterations, and the arithmetic used 20-bit multipliers. The spectrum of the oscillator's complex sinusoid stabilized with the AGC mechanism in (7) is shown in Figure 3(a), which also illustrates the ultra low phase noise of our oscillator. The AGC gain factor $g(n)$ in (7), less its nominal unity value, is shown in Figure 3(b).

An interesting aspect of the recursive CORDIC is that, for a fixed frequency sinusoid, the angle accumulator is initialized with the same angle value for each successive time sample. Thus, the sequence of add-subtract iterations in the CORDIC is identical for each computed trigonometric sample. The memory of the recursive CORDIC resides in the network states rather than in the traditional phase accumulator that forms and presents a sequence of phase angles modulo 2π to the CORDIC's angle accumulator. Thus, the phase error sequence is a constant for the recursive CORDIC; it is always the same angle error residing in the angle accumulator. Consequently, there is no

line structure in the spectrum of the recursive CORDIC, and the phase error correction is not applied to suppress phase error artifacts but rather to complete the phase rotation left incomplete due to the residual phase term in the angle accumulator. This is a very different DDS!

IMPLEMENTATION

As a practical note, there are truncating quantizers between the AGC multipliers and the feedback delay element registers. As such, the truncation error circulates in the registers and contributes an undesired dc component to the complex sinusoid output. This dc component can (and should) be suppressed by using a sigma delta-based dc cancellation loop between the AGC multipliers and the feedback delay elements [6].

CONCLUSIONS

We modified the traditional recursive DDS complex oscillator structure to a tangent/cosine configuration. The tan(θ) computations were implemented by CORDIC rotations avoiding the need for multiply operations. To minimize output phase angle error, we applied a post-CORDIC clean-up angle rotation. Finally, we stabilized the DDS output amplitude by an AGC loop. The phase-noise performance of the DDS is quite remarkable and we invite you, the reader, to take a careful look at its structure. A MATLAB-code implementation of the DDS is available at http://apollo.ee.columbia.edu/spm/?i=external/tipsandtricks.

ACKNOWLEDGMENT

Thanks to Rick Lyons for patience and constructive criticism above and beyond the call of duty.

AUTHOR

Fred Harris (fred.harris@sdsu.edu) teaches DSP and modem design at San Diego State University. He holds 12 patents on digital receivers and DSP technology. He has written over 140 journal and conference papers and is the author of the book *Multirate Signal Processing for Communication Systems* (Prentice Hall Publishing).

REFERENCES

[1] C. Dick, F. Harris, and M. Rice, "Synchronization in software defined radios—Carrier and timing recovery using FPGAs," in *Proc. IEEE Symp. Field-Programmable Custom Computing Machines*, Napa Valley, CA, pp. 195–204, Apr. 2000.

[2] J. Valls, T. Sansaloni, A. Perez-Pascual, V. Torres, and V. Almenar, "The use of CORDIC in software defined radios: A tutorial," *IEEE Commun. Mag.*, vol. 44, no. 9, pp. 46–50, Sept. 2006.

[3] F. Harris, C. Dick, and R. Jekel, "An ultra low phase noise DDS," presented at Software Defined Radio Forum Tech. Conf. (SDR-2006), Orlando FL, Nov. 2006.

[4] R. Lyons, *Understanding Digital Signal Processing*, 2nd ed. Upper Saddle River, NJ: Prentice Hall, pp. 576–578, 2004.

[5] C. Turner, "Recursive discrete-time sinusoidal oscillators," *IEEE Signal Processing Mag.*, vol. 20, no. 3, pp. 103–111, May 2003.

[6] C. Dick and F. Harris, "FPGA signal processing using sigma-delta modulation," *IEEE Signal Processing Mag.*, vol. 17, no. 1, pp. 20–35, Jan. 2000. SP

lecture NOTES continued from page 120

An image acquired with the single-pixel camera using about 60% fewer random measurements than reconstructed pixels is illustrated in Figure 3(c); compare to the target image in Figure 3(b). The reconstruction was performed via a total variation optimization [1], which is closely related to the ℓ_1 reconstruction in the wavelet domain. In addition to requiring fewer measurements, this camera can image at wavelengths where is difficult or expensive to create a large array of sensors. It can also acquire data over time to enable video reconstruction [10].

CONCLUSIONS: WHAT WE HAVE LEARNED

Signal acquisition based on compressive sensing can be more efficient than traditional sampling for sparse or compressible signals. In compressive sensing, the familiar least squares optimization is inadequate for signal reconstruction, and other types of convex optimization must be invoked.

ACKNOWLEDGMENTS

This work was supported by grants from NSF, DARPA, ONR, AFOSR, and the Texas Instruments (TI) Leadership University Program. Special thanks are due to TI for the DMD array used in the single-pixel camera. Thanks also to the Rice DSP group and Ron DeVore for many enlightening discussions and Justin Romberg for help with the reconstruction in Figure 3.

AUTHOR

Richard G. Baraniuk (richb@rice.edu) is the Victor E. Cameron Professor of Electrical and Computer Engineering at Rice University. His research interests include multiscale analysis, inverse problems, distributed signal processing, and sensor networks. He is a Fellow of the IEEE.

REFERENCES

Additional compressive sensing resources are available at dsp.rice.edu/cs.

[1] E. Candès, J. Romberg, and T. Tao, "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information," *IEEE Trans. Inform. Theory*, vol. 52, no. 2, pp. 489–509, Feb. 2006.

[2] D. Donoho, "Compressed sensing," *IEEE Trans. Inform. Theory*, vol. 52, no. 4, pp. 1289–1306, Apr. 2006.

[3] S. Mallat, *A Wavelet Tour of Signal Processing*. New York: Academic, 1999.

[4] R.G. Baraniuk, M. Davenport, R. DeVore, and M.B. Wakin, "A simple proof of the restricted isometry principle for random matrices (aka the Johnson-Lindenstrauss lemma meets compressed sensing)," *Constructive Approximation*, 2007 [Online]. Available: http://dsp.rice.edu/cs/jlcs-v03.pdf

[5] D. Baron, M.B. Wakin, M. Duarte, S. Sarvotham, and R.G. Baraniuk, "Distributed compressed sensing," 2005 [Online]. Available: http://dsp.rice.edu/cs/DCS112005.pdf

[6] J. Tropp and A.C. Gilbert, "Signal recovery from partial information via orthogonal matching pursuit," Apr. 2005 [Online]. Available: http://www-personal.umich.edu/~jtropp/papers/TG06-Signal-Recovery.pdf

[7] J. Haupt and R. Nowak, "Signal reconstruction from noisy random projections," *IEEE Trans. Inform. Theory*, vol. 52, no. 9, pp. 4036–4048, Sept. 2006.

[8] S. Kirolos, J. Laska, M. Wakin, M. Duarte, D. Baron, T. Ragheb, Y. Massoud, and R.G. Baraniuk, "Analog-to-information conversion via random demodulation," in *Proc. IEEE Dallas Circuits Systems Workshop*, Oct. 2006, pp. 71-74.

[9] M. Vetterli, P. Marziliano, and T. Blu, "Sampling signals with finite rate of innovation," *IEEE Trans. Signal Processing*, vol. 50, no. 6, pp. 1417–1428, June 2002.

[10] D. Takhar, V. Bansal, M. Wakin, M. Duarte, D. Baron, J. Laska, K.F. Kelly, and R.G. Baraniuk, "A compressed sensing camera: New theory and an implementation using digital micromirrors," in *Proc. Comput. Imaging IV SPIE Electronic Imaging*, San Jose, Jan. 2006. SP