Lionel Cordesses

Direct Digital Synthesis: A Tool for Periodic Wave Generation (Part 1)

Oscillators are the subject of intensive research. From Colpitts oscillators [1, chap. 7] to phase locked-loops [2], methods have been proposed to improve stability, frequency resolution, and spectral purity. Among the all-digital approaches such as the one presented in [3], direct digital frequency synthesis (referred to here as DDS) appeared in 1971 [4]. Three years later, this technique was embedded in a commercial unit measuring group delay of telephone lines [5]. DDSs are now available as integrated circuits and they output waveforms up to hundreds of megahertz.

While DDS is slowly gaining acceptance in new system designs, methods used to improve the quality of the generated waveform are seldom used, even nowadays. The purpose of Part 1 of this article is to give an overview of the basics of DDS, along with simple formulas to compute bounds of the signal characteristics. Moreover, several methods—some patented—are presented to overcome some of the limits of the basic DDS with a focus on improving output signal quality.

An Overview of DDS

The digital signal processing operation we want to perform is to generate a periodic, discrete-time waveform of known frequency F_o. The waveform may be a sinewave, as in [3]. It can also be a saw-tooth wave, a triangle wave, a square wave, or any periodic waveform. We will assume that the sampling frequency F_s is known and constant. Before proceeding with the theory of operation, we summarize why DDS is a valuable technique.

▲ 1) The tuning resolution can be made arbitrarily small to satisfy almost any design specification.

▲ 2) The phase and the frequency of the waveform can be controlled in one sample period, making phase modulation feasible.

▲ 3) The DDS implementation relies upon integer arithmetic, allowing implementation on virtually any microcontroller.

▲ 4) The DDS implementation is always stable, even with finite-length control words. There is no need for an automatic gain control.

▲ 5) The phase continuity is preserved whenever the frequency is changed (a valuable tool for tunable waveform generators).

Theory of Operation and Implementation

The implementation of DDS is divided into two distinct parts as shown in Figure 1: a discrete-time *phase generator* (the accumulator) outputting a phase value *ACC*, and a *phase to waveform converter* outputting the desired DDS signal.

From a Sampling Frequency to a Phase

The implementation of the DDS relies upon integer arithmetic. The size of the accumulator (or word length) is N b. Assuming that the period of the output signal is 2π rad, the maximum phase is represented by the integer number 2^N. Let us denote Δ_{ACC} the phase increment related to the desired output F_o frequency. It is coded as an integer number with $N-1$ b.

During one sample period T_s, the phase increases by Δ_{ACC}. It thus takes T_o to reach the maximum phase 2^N:

$$T_o = \frac{1}{F_o} = \frac{2^N T_s}{\Delta_{ACC}}. \quad (1)$$

We can rewrite (1) in terms of frequency F_o, as a function of Δ_{ACC}:

$$F_o = F_o(\Delta_{ACC}) = \frac{F_s}{2^N}\Delta_{ACC}. \quad (2)$$

The phase increment Δ_{ACC}, round-

"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. Contact Associate Editors Rick Lyons (r.lyons@ieee.org) or Amy Bell (abell@vt.edu).

1053-5888/04/$20.00©2004IEEE

ed to the nearest integer ($\lfloor x \rfloor$ is the integer part of x), is given by

$$\Delta_{\mathrm{ACC}} = \left\lfloor F_{\mathrm{o}} \frac{2^N}{F_{\mathrm{s}}} + 0.5 \right\rfloor. \quad (3)$$

Equation (2) is the basic equation of any DDS system. One can infer from (2) the tuning step $\Delta F_{\mathrm{o_{min}}}$, which is the smallest step in frequency, that the DDS can achieve (remember that Δ_{ACC} is an integer)

$$\begin{aligned} \Delta_{F_{\mathrm{Omin}}} &= F_{\mathrm{o}}(\Delta_{\mathrm{ACC}} + 1) \\ &\quad - F_{\mathrm{o}}(\Delta_{\mathrm{ACC}}) \\ &= \frac{F_{\mathrm{s}}}{2^N}(\Delta_{\mathrm{ACC}} + 1 - \Delta_{\mathrm{ACC}}) \\ &= \frac{F_{\mathrm{s}}}{2^N}. \end{aligned} \quad (4)$$

Equation (4) allows the designer to choose the number of bits (N) of the accumulator ACC. This number N is often referred to as the frequency tuning word length [6]. It is reckoned thanks to:

$$N = \left\lfloor \log_2 \left(\frac{F_{\mathrm{s}}}{\Delta_{F_{\mathrm{Omin}}}} \right) + 0.5 \right\rfloor. \quad (5)$$

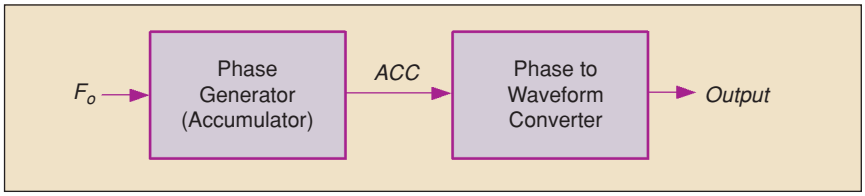
The minimum frequency $F_{\mathrm{o_{min}}}$ the DDS can generate is given by (2) with $\Delta_{\mathrm{ACC}} = 1$, the smallest phase increment which still increases the phase ($\Delta_{\mathrm{ACC}} = 0$ does not increase the phase). $F_{\mathrm{o_{min}}}$ is

$$F_{\mathrm{o_{min}}} = \frac{F_{\mathrm{s}}}{2^N}. \quad (6)$$

The maximum frequency $F_{\mathrm{o_{max}}}$ the DDS can generate is given by the uniform sampling theorem (Nyquist, Shannon, see, for instance, [7, chap. 9]):

$$F_{\mathrm{o_{max}}} = \frac{F_{\mathrm{s}}}{2}. \quad (7)$$

From a practical point of view a lower $F_{\mathrm{o_{max}}}$ is often preferred, $F_{\mathrm{o_{max}}} = F_{\mathrm{s}}/4$ for example. The lower that $F_{\mathrm{o_{max}}}$ is, the easier the analog reconstruction using a low-pass filter.



▲ *1. Fundamental DDS process.*

From a Phase to a Waveform

The phase is coded with N b in the accumulator. Thus, the waveform can be defined with up to 2^N phase values. In case 2^N is too large for a realistic implementation, the phase-to-amplitude converter uses fewer bits than N. Let us note P as the number of bits used as the phase information (with $P \leq N$). The output waveform values can be stored in a lookup table (LUT) with 2^P entries: the output value is computed as Output = LUT (ACC), which is implemented in the phase to waveform converter in Figure 1; other output waveform generation techniques, based upon approximations, are presented later.

DDS can generate a sinewave with an offset b and a peak amplitude a. The content of the LUT, containing the DDS output values, is computed for the index i ranging from 0 to $(2^P - 1)$ using

$$\mathrm{LUT}(i) = \left\lfloor (b + a\sin\left(\frac{2\pi i}{2^P}\right) + 0.5 \right\rfloor. \quad (8)$$

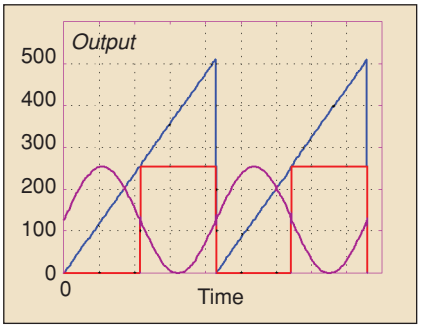
Using the LUT computed for $P = 9$, $a = 127.5$, and $b = 127.5$, the output waveform for $F_{\mathrm{s}} = 44,100$ Hz and $F_{\mathrm{o}} = 233$ Hz is plotted as the black curve in Figure 2.

One might want to generate two quadrature signals: one just has to read both LUT(i) and LUT $(i + 2^P/4)$, which, in turn, correspond to the sine and to the cosine functions.

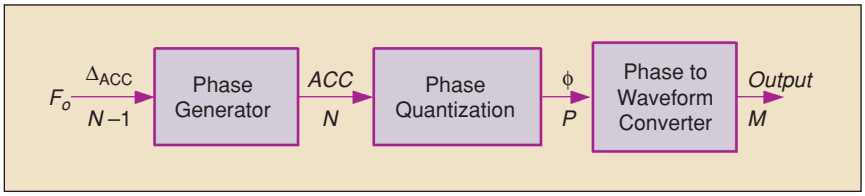
A square wave can be had with no computational overhead because that waveform is already available as the most significant bit of the phase accumulator ACC, as shown by the red curve in Figure 2. The most significant bit toggles every π rad, since the accumulator represents 2π rad. We must point out that this square wave is corrupted by phase jitter [8] of one sampling period T_{s}. This phase jitter is caused by the sampling scheme used to synthesize the waveform. To quote:

> ... the output of the direct digital synthesizer can occur only at a clock edge. If the output frequency is not a direct submultiple of the clock, a phase error between the ideal output and the actual output slowly increases (or decreases) until it reaches *one clock period*, at which time the error returns to zero and starts to increase (or decrease) again. [34]

A sawtooth signal is also available with no computational overhead. The linearly increasing phase accumulator ACC value is stored



▲ *2. Signals generated by software DDS: sinewave, squarewave, and sawtooth signals.*



▲ 3. Signals generated by software DDS: sinewave, squarewave, and sawtooth signals.

modulo 2^N, thus leading to a sawtooth signal as shown by the blue curve in Figure 2. The LUT is not used in this case, or it is the identity function: Output=ACC. With the use of logic gates, a triangular output waveform can be generated from the sawtooth.

Quantization Effects

Quantization occurs on both the ACC phase information and on the Output amplitude information. The DDS is now redrawn including this effect. The number of bits used by each variable is written below the variables on Figure 3.

Phase Quantization

Phase quantization occurs when the phase information ACC is truncated from N to P b as shown in Figure 3. The reason behind this quantization is to keep the memory requirements of the phase to waveform converter quite low: When implemented as a LUT, the size of the memory is $2^P \times M$ b. A realistic value for N is 32, but this would lead to a $2^{32} \times M$ memory that is not realistic. Thus we quantize the phase information Φ to P b, as it decreases the number of entries of the LUT.

Unfortunately, the phase quantization introduces noise on the phase signal Φ. It leads to *phase noise* (see [1, chap. 7] and [9, chap. 3]) and it produces unwanted spurious spectral components in the DDS output signals, often referred to as *spurs*. The difference between the carrier level (which is the desired signal) and the maximum level of spurs is called *spurious free dynamic range* (SFDR). A simplified formula given in [10] to estimate the maximum level of the spurs $S_{\max}$ when the carrier level is 0 dB is:

$$S_{\max} = -\text{SFDR} = -6.02P + 3.92\,\text{dB}. \quad (9)$$

For a detailed derivation of the exact formulas (including the frequency and the SFDR of spurs), the reader is referred to [11] and [12].

Amplitude Quantization

The output of the phase to waveform converter is quantized to M b, with M being the word length of the Output amplitude word. This quantization results in a signal-to-noise ratio (in this case, it is a noise-to-signal ratio) [10] usually approximated by:

$$SNR = -6.02M - 1.76\,\text{dB}. \quad (10)$$

This bound also limits the performance of the DDS, as the output spectrum will exhibit a $-6.02\,M - 1.76$ dB noise floor. Thanks to (9) and (10) one can infer (see [10]):

$$P = M + 1. \quad (11)$$

Thus, $S_{\max} = -6.02\,(M+1) + 3.992\,\text{dB} = -6.02M - 2.028$ dB and $\text{SNR} = -6.02M - 1.76$ dB leading to $S_{\max} < \text{SNR}$. This inequality means that the unwanted signals are caused by the amplitude quantization and not by the phase truncation. Knowing (11), we can now focus on improving the SFDR of a DDS.

Improving SFDR by Sinewave Compression

There are many techniques to improve the SFDR of a DDS. The easiest one would be to increase the phase word length. Thanks to (9) and (10), one can infer we can increase P (and thus M according to 11) to meet the technical specifications. The only drawback of this approach is the total amount of LUT memory, $2^P \times M$ b. For small P (such as $P = 9$ b and $M = 8$ b), implementing the LUT with a memory leads to simple, low-cost, hardware; see [13] and [8] for a realization based upon this method.

For higher values of P, the memory requirements become impractical at high frequency or for embedded system implementations. To circumvent this impediment, the solution is to compress the sine waveform, thus reducing memory consumption. Two methods are reported in the next sections. One is based upon symmetry and the other on sinewave approximations.

A Quarter of a Sinewave

Instead of storing the entire sinewave $f(\Phi) = \sin(\Phi)$ for $0 \le \Phi \le 2\pi$, one can store the same function for $0 \le \Phi \le \pi/2$ and use symmetry to get the complete 2π waveform range. This approach only uses 2^{P-2} entries in the LUT, leading to a LUT-size compression ratio of 4:1. The full sinewave can be reconstructed at the expense of some hardware (see [5], [14] and [9]). From here out, we will only deal with a quarter of a sinewave. Next we discuss four methods of approximating a sinewave.

Sinewave Approximations

The first sinewave approximation method goes as follows: instead of storing $f(\Phi) = \sin(\Phi)$ using M b, one can store $g(\Phi) = \sin(\Phi) - 2\Phi/\pi$, hence the name *sine-phase difference algorithm* found in [14].

It has been shown in [14] that this new function g only needs $M-2$ b to get the same amplitude quantization for the sinewave (see Figure 4 for an example). The only drawback is the need for an adder at the output of the LUT.

The second sinewave approximation method is called the *Sunderland technique*. This method, named after its author [15], makes use of trigonometric identities. It has been used for $P=12$, and it uses the following identity:

$$\begin{aligned}\sin(A+B+C) = \sin(A+B)\cos(C) \\ + \cos(A)\cos(B) \\ \times \sin(C) - \sin(A) \\ \times \sin(B)\sin(C).\end{aligned} \quad (12)$$

The 12 b of the phase are:

- A, the four most significant bits (with $0 \le A \le \pi/2$)
- B, the following four bits [with $0 \le B \le (\pi/2)/(2^4)$]
- C, the four least significant bits [with $0 \le C \le (\pi/2)/(2^8)$].

Equation (12) is then approximated by

$$\begin{aligned}\sin(A+B+C) \approx \sin(A+B) \\ + \cos(A)\sin(C).\end{aligned} \quad (13)$$

Using two LUTs [one for $\sin(A+B)$ and one for $\cos(A)\sin(C)$] leads to a significant amount of compression. The $\sin(A+B)$ LUT uses $2^8 \times 11$ b ($P=12$ thus $M=P-1=11$). The second LUT is filled with small numbers, thus requiring less than M b (actually 4 b; see [15]). Finally, the compression ratio of this architecture is 51:1 (see [16] for a comparison of various compression methods).

Several improvements to this architecture have been presented (see [14]) and the compression ratio of the modified Sunderland technique leads to a 59:1 compression ratio [16]. The same method has been used in [17] with a 128:1 compression ratio and in [18] with a 165:1 compression ratio.

The third sinewave approximation method involves first-order Taylor series expansions. Let us introduce δ_Φ with $\delta_\Phi << \Phi$. The Taylor series expansion of the sine function is

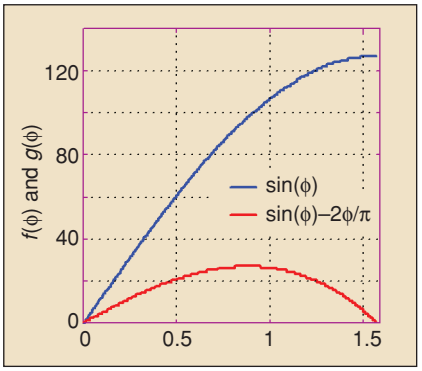
$$\sin(\Phi+\delta_\Phi) \approx \sin(\Phi) + \delta_\Phi \cos(\Phi). \quad (14)$$

Instead of storing the sine function, the key idea presented in [19] and described in [20] proposes to use two coarse LUTs storing $\sin(\Phi)$ and $\cos(\Phi)$. Moreover, the sine phase difference algorithm can be used to store efficiently $\sin(\Phi)$, further decreasing the size of the LUT.

The compression ratio obtained with this method is 64:1 [19] and even reaches 67:1 [16]. Another method has been introduced in [21] where the sine function is approximated by linear interpolation. A few samples (16 in [22]) of the sine function are stored in a LUT, and the values are computed using linear interpolation. The compression ratio, computed thanks to data given in [22], is: $10 \times 2^{11}/960 \approx 21{:}1$. Another implementation of the linear solution does not rely on a LUT. Using notations as in [23], the sine function is written as

$$\sin(\Phi+\delta_\Phi) \approx y_0 + m_0(\Phi-\delta_\Phi). \quad (15)$$

The solutions presented in [24] and used in [23] carefully impose a power of two number of segments, thus using the most significant bits of δ_Φ as an address. Moreover, to further decrease complexity, the length of all the segments are equal. The implementation of (15) only uses one multiplication and one addition, without any complex address decoder. There is no LUT in this design. According to their authors, such an approach reaches the performance of other methods, showing a 60 dBc SFDR for $P=12$ b phase [23]. The method is now patented [25].



▲ *4. Sine-phase difference LUT example (P = 9, M = 8).*

The fourth sinewave approximation method involves higher-order Taylor series expansions. In [26], the Taylor series expansion of the sine function is

$$\begin{aligned}\sin(\Phi+\delta_\Phi) \approx \sin(\Phi) + \delta_\Phi\cos(\Phi) \\ - \frac{1}{2}(\delta_\Phi)^2\sin(\Phi).\end{aligned} \quad (16)$$

The compression ratio, given in [27], is 110:1 for $P=12$ b and a SFDR of 85 dBc.

Higher-order interpolation is also used for sinewave compression: parabolic interpolation is presented in [28]. Only interpolation coefficients V_1, V_2, and V_3 are stored in the LUT, and the value of the sine function is reckoned thanks to:

$$\begin{aligned}\sin(\Phi+\delta_\Phi) \approx V_1(\Phi) + \delta_\Phi V_2(\Phi) \\ + (\delta_\Phi)^2 V_3(\Phi).\end{aligned} \quad (17)$$

The compression ratio obtained using (17) is given in [28] for a 64 dBc SFDR at $M=11$ b is 157:1. As in the previous first-order Taylor series method, there is a counterpart of the LUT-less method. Based upon a quadruple angle equality:

$$\begin{aligned}\cos(4\Phi) = 1 - 8\sin^2(\Phi) \\ \times [1-\sin^2(\Phi)].\end{aligned} \quad (18)$$

with $0 \le \Phi \le \pi/8$, (18) it is approximated in [29] by

$$\cos(4\Phi) \approx 1 - 8\Phi^2(1 - \Phi^2). \quad (19)$$

Equation (19) is implemented with multipliers and adders only.

There are still other approaches to approximating a sinewave. A phase to sinewave converter can be implemented thanks to an angle rotation algorithm, such as the CORDIC (COordinate Rotation DIgital Computer) algorithm [30]. A DDS based on this method is described in [31], and the effect of finite precision on the CORDIC converter is analyzed in [32]. Another method relies upon a non-linear digital to analog converter that implements the sinewave generation [33].

There are tradeoffs with each of the above sinewave approximation techniques, and no single technique is best for all DDS applications.

In Part 2 of this article, we will present additional tricks used to maximize DDS SFDR.

Acknowledgments

I would like to thank Rick Lyons who helped improve this article through many comments and a thorough reading of the original manuscript.

Lionel Cordesses is a control engineer at the Technocentre Renault (Guyancourt, France). He is a member of the IEEE and of the ACM.

References

[1] U.L. Rohde, J. Whitaker, and T.T.N. Bucher, *Communications Receivers,* 2nd ed. New York: McGraw Hill, 1997.

[2] U.L. Rohde, *Digital PLL Frequency Synthesizers—Theory and Design*. Englewood Cliffs, NJ: Prentice-Hall, 1983.

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

[4] J. Tierney, C. Rader, and B. Gold, "A digital frequency synthesizer," *IEEE Trans. Audio Electroacous.*, vol. 19, no. 1, pp. 48–57, Mar. 1971.

[5] D.H. Guest, "Simplified data-transmission channel measurements," *Hewlett-Packard J*, vol. 26, no. 3, pp. 15–24, Nov. 1974.

[6] "A technical tutorial on digital signal synthesis," Analog Devices, Inc., Tech. Rep., 1999.

[7] H.J. Blinchikoff and A.I. Zverev, *Filtering in the Time and Frequency Domains.* New York: Noble, 2001.

[8] E. McCune, "Create signals having optimum resolution, response, and noise," *EDN*, vol. 36, no. 6, pp. 95–108, Mar. 14, 1991.

[9] J.A. Crawford, *Frequency Synthesizer Design Handbook.* Norwood, MA: Artech House, 1994.

[10] P. O'Leary and F. Maloberti, "A direct-digital synthesizer with improved spectral performance," *IEEE Trans. Commun.*, vol. 39, no. 7, pp. 1,046–1,048, July 1991.

[11] V.F. Kroupa, V. Cizek, J. Stursa, and H. Svandova, "Spurious signals in direct digital frequency synthesizers due to the phase truncation," *IEEE Trans. Ultrason., Ferroelect. Freq. Contr.*, vol. 47, no. 5, pp. 1,166–1,172, Sept. 2000.

[12] F. Curticapean and J. Niittylahti, "Exact analysis of spurious signals in direct digital frequency synthesizers due to phase truncation," *Electron. Lett.*, vol. 39, no. 6, pp. 499–501, Mar. 2003.

[13] B-G. Goldberg, "Digital frequency synthesizer," U.S. Patent 4 752 9029, July 1985.

[14] H.T. Nicholas, III, H. Samueli, and B. Kim, "The optimization of direct digital frequency synthesizer performance in the presence of finite word length effects," in *Proc. 42nd Annu. Frequency Control Symp.*, June 1988, pp. 357–363.

[15] D.A. Sunderland, R.A. Strauch, S.S. Wharfield, H.T. Peterson, and C. R. Cole, "CMOS/SOS frequency synthesizer LSI circuit for spread spectrum communications," *IEEE J. Solid-State Circuits,* vol. 19, no. 4, pp. 497–506, Aug. 1984.

[16] K.A. Essenwanger and V.S. Reinhardt, "Sine output DDSs. A survey of the state of the art," in *Proc. 1998 IEEE Int. Frequency Control Symp.*, May 1998, pp. 370–378.

[17] H.T. Nicholas III and H. Samueli, "A 150-MHz direct digital frequency synthesizer in 1.25-mm CMOS with -90-dBc spurious performance," *IEEE J. Solid-State Circuits*, vol. 26, no. 12, pp. 1959–1969, Dec. 1991.

[18] G.W. Kent and N.-H. Sheng, "A high purity, high speed direct digital synthesizer," in *Proc. 1995 IEEE Int. 49th. Frequency Control Symp.*, June 1995, pp. 207–211.

[19] "DDS tutorial," Scitech Electronics, Inc, San Diego, CA, Tech. Rep. V3, 1991.

[20] B-G. Goldberg, "Source of quantized samples for synthesizing sinewaves," U.S. Patent 5 321 642, June 1994.

[21] R.A. Freeman, "Digital sine conversion circuit for use in direct digital synthesizers," U.S. Patent 4 809 205, Nov. 1989.

[22] A. Bellaouar, M. Obrecht, A. Fahim, and M.I. Elmasry, "A low-power direct digital frequency synthesizer architecture for wireless communications," in *Proc. IEEE 1999 Custom Integrated Circuits*, May 1999, pp. 593–596.

[23] J.M.P. Langlois and D. Al-Khalili, "A low power direct digital frequency synthesizer with 60 dBc spectral purity," in *Proc. 12th ACM Great Lakes Symp. VLSI*, 2002, pp. 166–171.

[24] J.M.P. Langlois and D. Al-Khalili, "Piecewise continuous linear interpolation of the sine function for direct digital frequency synthesis," in *IEEE Radio Frequency Integrated Circuits (RFIC) Symp.*, June 2003, pp. 579–582.

[25] D. Al-Khalili and J.M.P. Langlois, "Phase to sine amplitude conversion system and method," European Patent 1286258, Feb. 2003.

[26] L.A. Weaver, Jr, and R.J. Kerr, "High resolution phase to sine amplitude conversion," U.S. Patent 4 905 177, Jan. 1988.

[27] J. Vankka, "Methods of mapping from phase to sine amplitude in direct digital synthesis," *IEEE Trans. Ultrason., Ferroelect. Freq. Contr.*, vol. 44, no. 2, pp. 526–534, Mar. 1997.

[28] A.M. Eltawil and B. Daneshrad, "Piece-wise parabolic interpolation for direct digital frequency synthesis," in *Proc. IEEE Custom Integrated Circuits Conf.*, May 2002, pp. 401–404.

[29] C.-C. Wang, H.-C. She, and R. Hu, "A ROM-less direct digital frequency synthesizer by using trigonometric quadruple angle formula," in *Proc. 9th Int. Conf. Electronics, Circuits and Systems*, Sept. 2002, vol. 1, pp. 65–68.

[30] G.C. Gielis, R. van de Plassche, and J. van Valburg, "A 540-MHz 10-b polar-to-cartesian converter," *IEEE J. Solid-State Circuits*, vol. 26, no. 11, pp. 1645–1650, Nov. 1991.

[31] A. Madisetti, A.Y. Kwentus, and A.N. Willson Jr, "A 100-MHz, 16-b, direct digital frequency synthesizer with a 100-dBc spurious-free dynamic range," *IEEE J. Solid-State Circuits*, vol. 34, no. 8, pp. 1034–1043, Aug. 1999.

[32] C.Y. Kang and E.E. Swartzlander, Jr,. "An analysis of the CORDIC algorithm for direct digital frequency synthesis," in *Proc. IEEE Int. Conf. Application-Specific Systems, Architectures and Processors,* July 2002, pp. 111–119.

[33] A. McEwan and S. Collins, "Analogue interpolation based direct digital frequency synthesis," in *Proc. 2003 Int. Symp. Circuits and Systems ISCAS '03,* May 2003, vol. 1, pp. 621–624.

[34] C.E. Wheatley III and D.E. Phillips, "Spurious suppression in direct digital synthesizers," in *Proc. 35th Annu. Frequency Control Symp.*, May 1981, pp. 428–435.