Piecewise Continuous Linear Interpolation of the Sine Function for Direct Digital Frequency Synthesis

J.M.P. Langlois and D. Al-Khalili

Department of Electrical and Computer Engineering
Royal Military College of Canada, Kingston (Ontario), Canada
pierre.langlois@rmc.ca, alkhalili-d@rmc.ca

Abstract – **This paper discusses the design of Direct Digital Frequency Synthesizers (DDFS) based on the linear interpolation of the sine function. The problem of approximating the sine function within a desired error bound is specifically considered. The use of linear segments is favorable for hardware implementation because of the low processing complexity requirements. A relation between the minimum number of linear segments, the resolution with which segment slopes are expressed, and the achievable precision is derived. Tradeoffs between memory storage requirements and computational complexity are identified, and architectural and implementation issues are discussed. Example designs achieving 8, 10 and 12 bits of amplitude resolution with 59, 77 and 86 dBc of Spurious Free Dynamic Range (SFDR) are presented.**

I. Introduction

The flexibility and performance characteristics of Direct Digital Frequency Synthesis (DDFS) make this kind of synthesizer very attractive for reconfigurable communications equipment and software radio applications. The combination of excellent frequency resolution and high frequency switching rates differentiate DDFS from other synthesizers based on phase locked loops. DDFS has been well described in the literature [1].

The basic DDFS architecture includes a phase accumulator and a Phase to Sinusoid Amplitude Converter (PSAC). The output frequency is given by:

$$f_{out} = f_0 \times \frac{FCW}{2^N} \tag{1}$$

where f_0 is the frequency of the clock reference, FCW is the frequency control word, and N is the width of the phase accumulator.

The frequency resolution of the synthesizer is given by the ratio of the clock reference to the number of states of the phase accumulator. Hence, a large N is often selected, at the expense of a potential exponential increase in the PSAC complexity. For this reason, only M most significant phase accumulator bits are retained. The quadrant symmetry of the sine function is also normally exploited to reduce the PSAC complexity by more than a factor of four. The resulting architecture is shown in Fig. 1.

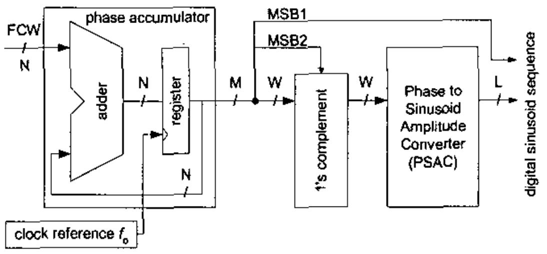

Fig. 1 Single phase DDFS with phase truncation and quadrant symmetry.

Several approaches have been proposed for the design of the PSAC, including using a ROM Look-Up Table (LUT), angular decomposition, sine amplitude compression, CORDIC and other angular rotation algorithms, and polynomial approximations [1]. All of these methods make a trade-off between computational complexity and memory storage. In most cases, the goal is to achieve a high Spurious Free Dynamic Range (SFDR) and a high maximum clock rate, while minimizing silicon area and power consumption. The SFDR is defined as the ratio of the power of the greatest undesired frequency spur to the power of the desired output frequency.

In this paper, we consider PSACs based on a first order polynomial approximation, i.e. making a linear interpolation of the sine function. This approach presents interesting hardware implementation advantages since computational complexity is limited to a single multiplication and an addition.

The paper is divided into 6 sections including this introduction. In section II, we review linear interpolation PSAC for DDFS. Section III discusses the problem of the linear interpolation of the sine function. System implementation issues are presented in section IV, and design examples are given in V. Conclusions are found in section VI.

II. Linear Interpolation PSAC for DDFS

To the best of our knowledge, Freeman [2] was the first to use a linear interpolation technique in DDFS. He effectively used 16 piecewise continuous linear segments to approximate the first quadrant of the sine function. Segment slopes and initial amplitudes were stored in two small ROMs, and a third ROM stored correction values. Multipliers and adders completed the architecture. Bellaouar et al. [3] used a 1st degree Taylor series expansion in 32 equal length segments, adding the capability of generating quadrature sinusoids. Liu et al. [4] decomposed the first quadrant of the sine function with linear segments of unequal lengths, whose number, slopes and y-intercepts were carefully selected to achieve a desired precision in the sine amplitude estimate. The work by Curticăpean et al. [5] can be seen as a combination of angular decomposition and linear interpolation. It features high SFDR, reduced ROM size, reasonable computational costs and reduced power consumption. El Said and Elmasry [6] improved on Bellaouar et al.'s work by shifting the interpolation point of the Taylor series expansion to the center of each interval. The result is a decrease in the required ROM storage at a slight increase in computational complexity.

As shown in Fig. 1, the role of the PSAC is to calculate an approximation of the sine function for first quadrant angles. Let x represent a scaled phase angle in the interval [0, 1[, then the output of the PSAC is given by

$$f(x) = A\sin(\frac{\pi x}{2}) - \varepsilon(x) \quad 0 \le x < 1 \tag{2}$$

where A is some amplitude equal to or less than 1, and ε is the approximation error. The approximated sine amplitude in the first quadrant is expressed with L bits, giving a synthesizer output of $(L + 1)$ bits. The amplitude factor A should be selected to maximize the synthesizer output amplitude without clipping. A reasonable choice is given by $(2^L - 1) / 2^L$.

For a linear interpolation DDFS, the PSAC implements equation (2) as follows:

$$f(x) = \begin{cases} y_0 + m_0(x - x_0) & x_0 \le x < x_1 \quad (x_0 = 0) \\ y_1 + m_1(x - x_1) & x_1 \le x < x_2 \\ \vdots & \\ y_k + m_k(x - x_k) & x_k \le x < x_{k+1} \\ \vdots & \\ y_{s-1} + m_{s-1}(x - x_{s-1}) & x_{s-1} \le x < x_s \quad (x_s = 1) \end{cases} \tag{3}$$

where s is the number of segments, m_k and y_k are a segment's slope and initial amplitude, respectively, and x_k is a segment's lower bound.

The selection of the number and length of the linear segments can greatly simplify the implementation of equation (3). If s is chosen to be a power of two, the $\log_2 s$ most significant bits of x give the segment number, k, and can directly address the LUTs storing the slopes and initial amplitudes. If the segments are equal in length, then the least significant $W - \log_2 s$ bits of x give the result of the $(x - x_k)$, and the segment bounds x_k are equal to k / s. Furthermore, the length of each segment is equal to $1 / s$.

The general architecture of a PSAC using linear interpolation and implementing these hardware optimizations is shown in Fig. 2. The scaled phase angle x is expressed with $W = M - 2$ fractional bits and the output is expressed with L fractional bits.

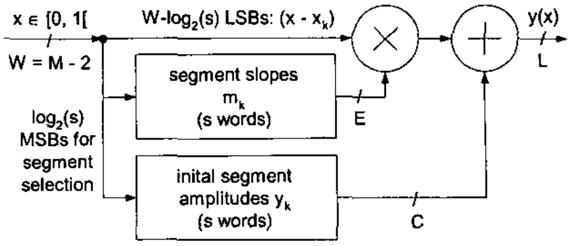

Fig. 2 Linear interpolation PSAC general architecture.

From Fig. 2, it can be seen that the implementation of the PSAC with a linear interpolation architecture has a very low computational complexity, when compared with other methods, since a single multiplication and an addition are required.

III. Linear Interpolation of The Sine Function

A. Error Analysis

In order to achieve the equivalent of R bits of amplitude resolution, the absolute value of the error $|\varepsilon(x)|$ in (2) must be less than ½LSB, or 2^{-R}. From equations (2) and (3), the interpolation error in segment k is given by

$$\varepsilon(x) = A\sin(\frac{\pi x}{2}) - (m_k(x - x_k) + y_k) \quad x_k \le x < x_{k+1} \tag{4}$$

The point x_{kp} where the error signal $\varepsilon(x)$ is greatest can be found by equating the first derivative of (4) to 0:

$$x_{kp} = \frac{2}{\pi}\cos^{-1}(\frac{2m_k}{A\pi}) \tag{5}$$

Depending on the value of m_k, there are two possible cases, shown in Fig. 3. For case a., x_{kp} lies inside segment k's bounds, and for case b. it doesn't. In the figure, the symbols ε_{kL}, ε_{kp}, and ε_{kR}, represent the error at the left bound, at the peak, and at the right bound of segment k:

$$\begin{aligned} \varepsilon_{kL} &= \varepsilon(x_k) = A\sin(\frac{\pi x_k}{2}) - (m_k(x_k - x_k) + y_k) \\ \varepsilon_{kp} &= \varepsilon(x_{kp}) = A\sin(\frac{\pi x_{kp}}{2}) - (m_k(x_{kp} - x_k) + y_k) \\ \varepsilon_{kR} &= \varepsilon(x_{k+1}) = A\sin(\frac{\pi x_{k+1}}{2}) - (m_k(x_{k+1} - x_k) + y_k) \end{aligned} \tag{6}$$

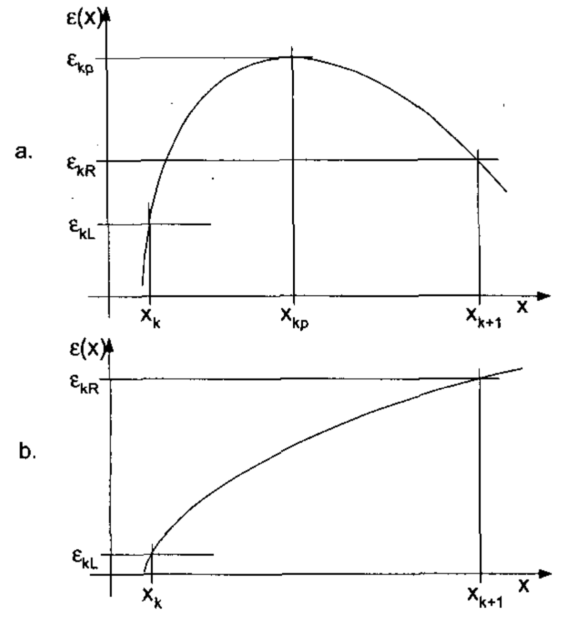

Fig. 3 a. Peak error lies inside segment bounds.
b. Peak error lies outside segment bounds.

B. Selection of Segment Coefficients

In order to minimize the absolute value of the error $|\varepsilon(x)|$ inside a segment, that segment's initial amplitude y_k must be selected so that the error signal $\varepsilon(x)$ is distributed evenly about the x axis. In case a., we should have:

$$|\varepsilon(x)|_{\max} = \varepsilon_{kp} = -\min(\varepsilon_{kL}, \varepsilon_{kR}), \quad (7)$$

and in case b. we should have:

$$|\varepsilon(x)|_{\max} = |\varepsilon_{kL}| = |\varepsilon_{kR}|. \quad (8)$$

Given a segment's width $x_{k+1} - x_k$ and an amplitude A, the minimum maximum error $|\varepsilon(x)|_{\max}$ on this segment will be attained for a specific situation in case a., i.e. $\varepsilon_{kL} = \varepsilon_{kR}$. From (6), the corresponding segment slope m_{kopt} is:

$$m_{kopt} = \frac{A}{x_{k+1} - x_k}\left(\sin(\frac{\pi x_{k+1}}{2}) - \sin(\frac{\pi x_k}{2})\right) \quad (9)$$

From these observations, the following coefficient selection algorithm therefore emerges once design parameters s, R, E, and C have been selected. For each segment:

1. calculate the value of m_{kopt} using (9);
2. obtain m_k by rounding m_{kopt} to E bits;
3. find the corresponding value of x_{kp} from (5);
4. calculate y_k according to one of the following equations:

$$y_k = \frac{1}{2}\left(A\sin(\frac{\pi x_{k+1}}{2}) - m_k(x_{k+1} - x_k) + A\sin(\frac{\pi x_k}{2})\right) \quad (10)$$

if case b. occurs, i.e. $x_{kp} < x_k$ or $x_{kp} > x_{k+1}$; or,

$$y_k = \frac{1}{2}\left(A\sin(\frac{\pi x_{kp}}{2}) - m_k(x_{kp} - x_k) + A\sin(\frac{\pi x_k}{2})\right) \quad (11)$$

if case a. occurs and if $m_k < m_{\text{kopt}}$ (and hence $\varepsilon_{kR} > \varepsilon_{kL}$); or,

$$\begin{aligned} y_k &= \frac{1}{2}(A\sin(\frac{\pi x_{kp}}{2}) - m_k(x_{kp} - x_k)) \\ &+ \frac{1}{2}(A\sin(\frac{\pi x_{k+1}}{2}) - m_k(x_{k+1} - x_k)) \end{aligned} \quad (12)$$

if case a. occurs and if $m_k > m_{\text{kopt}}$, (and hence $\varepsilon_{kR} < \varepsilon_{kL}$).

5. round y_k to C bits;
6. calculate the maximum amplitude error $|\varepsilon(x)|_{\max}$ according to (7) or (8), depending on the case.

C. Achievable Resolution Given s and E

The procedure described in the previous subsection can be followed to find the global maximum amplitude error for all segments. This was repeated for various values of the number of segments s and the number of bits E with which the slopes m_k are expressed. The results are shown in Fig. 4.

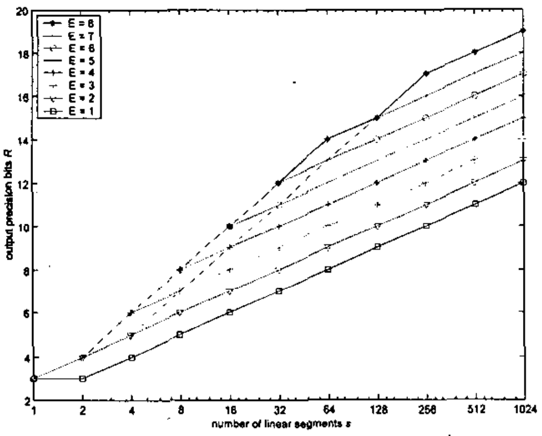

Fig. 4 Output resolution given s and E

From Fig. 4, it can be seen that there is a bound for which it is not possible to achieve a desired output resolution given a number of segments, regardless of the precision with which the m_k are expressed. It can also be seen that there is a linear relationship, passed a certain point, between the output resolution, the value of E, and the log in base 2 of the number of segments used. For example, to achieve 8 bits of amplitude resolution with 8 segments, at least 5 bits must be used to quantize the m_k. Achieving 12 bits of resolution with no more than 2 bits for the segment slopes requires at least 512 segments.

It must be noted that Fig. 4 gives the worst case requirements for all segments in a set. In actual fact, the resolution for some specific segments will be much greater. Some segments may thus be combined to reduce memory storage requirements. It must also be noted that Fig. 4 assumes a infinite value of C to remove the effect of the quantization of the y_k.

IV. Implementation Issues

Fig. 4 outlines the fact that a tradeoff can be made between the number of linear segments used and the resolution with which the segment slopes are expressed. Considering Fig. 2, it is obvious that both these parameters will affect the complexity of the multiplier and the total storage requirements.

Total storage in bits is given by $s \times (C + E)$. The complexity of the row decoders in LUTs is directly proportional to the number of rows, and hence this product is a good indicator of total storage cost.

The complexity of the multiplier can be approximated as the product of the length of its operands, in this case $(W - \log_2 s) \times E$. Hence, choosing a large s is favorable, since it reduces the minimum value of E required, and it also reduces the width of the operand $(x - x_k)$.

The size of the multiplier also affects the maximum clock rate of the system. If it is very large, pipelining registers may be required and they may significantly affect the total system complexity.

A tradeoff must therefore be made between the values of s and E, and it must be based on a comparison of the relative costs of implementing the multiplication operation and the storage cost per bit.

V. Design Examples and Simulation Results

Using the procedure described above, a simple design with $s = 8$ segments achieving 8 bits of resolution was produced. Its coefficients are given in Table 1 below. With a phase resolution $M = 11$ bits, this design achieves an SFDR of −59 dBc. For this design, the other parameters are $L = 9$, $E = 5$ and $C = 9$. Total storage is therefore equal to 112 bits. The multiplier size is 6×5.

A second design was produced. It achieves 10 bits of amplitude resolution with $s = 32$ segments. The other parameters are $L = 12$, $E = 5$, and $C = 10$. The SFDR is −76.7 dBc for a phase resolution of $M = 14$ bits. Total storage is equal to 480 bits. The multiplier size is 7×5.

A third design achieving 12 bits of resolution and 86 dBc of SFDR for a phase resolution of $M = 16$ bits was also considered. It uses $s = 64$ segments. System parameters are $L = 14$, $E = 6$ and $C = 12$. Total storage is equal to 1152 bits, and the multiplier size is 8×6.

TABLE I
SIMPLE DESIGN EXAMPLE

k	m_k	y_k
0	25 / 16	0 / 512
1	24 / 16	99 / 512
2	22 / 16	195 / 512
3	19 / 16	283 / 512
4	16 / 16	360 / 512
5	12 / 16	423 / 512
6	7 / 16	471 / 512
7	2 / 16	500 / 512

VI. Conclusion

In this paper, we have discussed the design of Direct Digital Frequency Synthesizers (DDFS) based on the linear interpolation of the sine function. The problem of approximating the sine function within a desired error bound was specifically considered. The use of linear segments was shown to be favorable for hardware implementation because of the low processing complexity requirements. A relation between the minimum number of linear segments, the resolution with which segment slopes are expressed, and the achievable precision was derived. Tradeoffs between memory storage requirements and computational complexity were identified, and architectural and implementation issues were discussed. Three example designs achieving 8, 10 and 12 bits of amplitude resolution were presented.

References

[1] V.F. Kroupa, Ed., *Direct Digital Frequency Synthesizers*, IEEE Press, 1999.

[2] R.A. Freeman, "Digital sine conversion circuit for use in direct digital synthesizers," U.S. Patent 4,809,205, 28th February 1989.

[3] A. Bellaouar, M. S. O'brecht, A. M. Fahim, and M. I. Elmasry, "Low-power direct digital frequency synthesis for wireless communications," *IEEE Journal of Solid-State Circuits*, vol. 35, no. 3, March 2000, pp. 385-390.

[4] S.-I. Liu, T.-B. Yu and H.-W. Tsao, "Pipeline direct digital frequency synthesiser using decomposition method," *IEE Proceedings on Circuits, Devices and Systems*, vol. 148, no. 3, June 2001, pp. 141-144.

[5] F. Curticăpean, K.I. Palomäki and J. Niittylahti, "Direct digital frequency synthesiser with high memory compression ratio," *Electronic Letters*, 11th October 2001, Vol. 37, No. 21, pp. 1275-1277.

[6] M.M El Said and M.I. Elmasry, "An improved ROM compression technique for direct digital frequency synthesizers," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, Phoenix AZ, 26-29 May 2002, pp. 437-440.