Library of Congress Cataloging-in-Publication Data
Crawford, James A.
Frequency synthesizer design handbook/James A. Crawford.
Includes bibliographical references and index.
ISBN 0-89006-440-7
1. Frequency synthesizers. I. Title
TK7872.F73C73 1994
621.3815'486–dc20

94-7655
CIP

In Figure 7.32, all parameters were left unchanged but $F_r = 640$ was selected to illustrate what can happen when $(F_r, 2^L) > 1$ occurs. In this case, the skewed spur levels indicate that elements of both AM and PM are present and the strongest spur component has increased to −60 dBc. This is substantially worse than that observed in the first two cases, which underscores the need to consider the worst case situation in actual DDS design.

7.3.5 Techniques for Mapping θ to Sin(θ)

The mapping technique [11] used for transforming the phase accumulator phase value θ to $\sin(\theta)$ is particularly crucial if table storage size is to be kept reasonable. Each additional bit used in the lookup table process potentially represents a doubling of the required table storage space.

The most simple means for compressed $\sin(\theta)$ storage is to exploit the symmetry of the function about $\pi/2$ and π. With proper manipulation of the phase and amplitude, lookup table samples need only be stored for phase values spanning the 0 to $\pi/2$ range. Such a scheme is shown schematically in Figure 7.33.

Normally, 2's-complement arithmetic is used in digital computing elements. However, this numerical representation presents some disadvantages wherever the negative of a number must be computed because formal negation involves first complementing each bit in the binary value followed by an addition of one to the quantity. In the table lookup case, if an LSB/2 offset is included in the number to be complemented, a simple 1's complementor can be used in place of the more complex 2's complementor without incurring additional error.

Sin(θ) Table Compression

One of the earliest methods used for sine table compression over the 0 to $\pi/2$ range is that of Sunderland et al. [12]. The subject architecture for this compression method is shown in Figure 7.34. In short, this technique allows the otherwise long table lookup ROM with 2^{A+B+C} storage locations to be replaced by two smaller ROMs having storage sizes of 2^{A+B} and 2^{A+C} locations. In the case where $A = B = C = 4$ and $D = 12$ bits, the noncompressed table would require 45056 bits (11×2^{12}). In contrast, even if full 12-bit-wide storage is used in both smaller ROMs, the total required bit storage is substantially less at 5632 bits ($11 \times 2^8 \times 2$) resulting in an 8:1 storage savings. The actual required word width for the second ROM in the present example is only 4 bits, which when factored in results in a total storage requirement of only 3840 bits ($11 \times 2^8 + 4 \times 2^8$), which is equivalent to an 11.73:1 storage reduction.

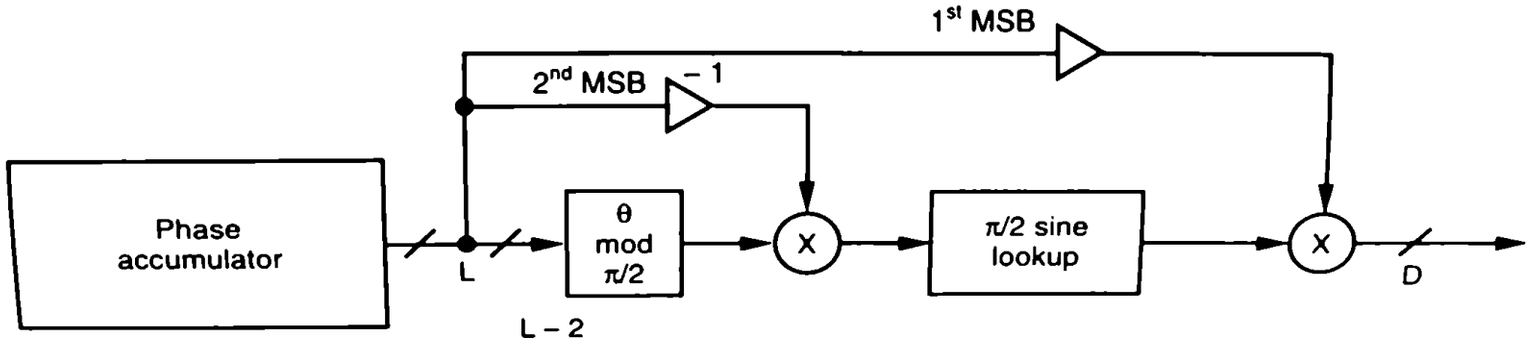

Figure 7.33 Exploitation of $\sin(\theta)$ symmetries about $\pi/2$ and π for table storage compression. (© 1988 IEEE; reprinted with permission). After: [11], Figure 6.

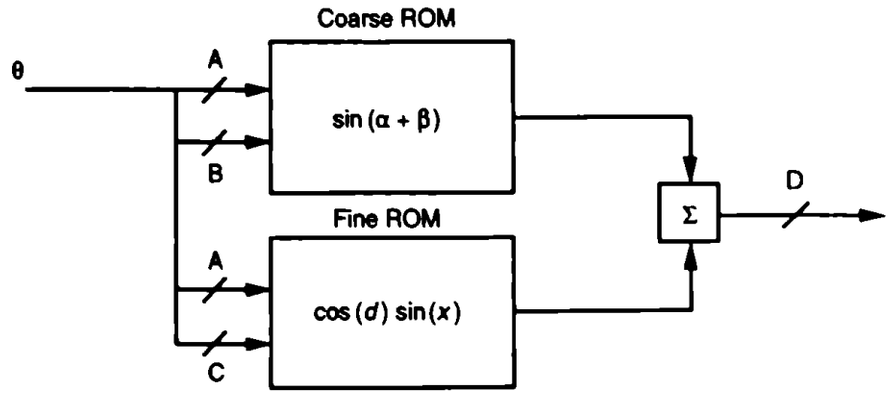

Figure 7.34 Sunderland architecture for compressed sin(θ) table storage. (© 1988 IEEE; reprinted with permission). After: [11], Figure 7.

The Sunderland technique is based on simple trigonometric identities. The phase accumulator value θ is first reflected into the first quadrant by computing $\varphi = \theta \bmod \pi/2$ and then further decomposed into a sum of three other angles as

$$\varphi = \alpha + \beta + \chi \tag{7.55}$$

where

$$\alpha < \frac{\pi}{2}$$

$$\beta < \frac{\pi}{2} 2^{-A} \tag{7.56}$$

$$\chi < \frac{\pi}{2} 2^{-(A+B)}$$

Using the trigonometric identity

$$\begin{aligned} \sin(\alpha + \beta + \chi) = {} & \sin(\alpha + \beta)\cos(\chi) \\ & + \cos(\alpha)\cos(\beta)\sin(\chi) - \sin(\alpha)\sin(\beta)\sin(\chi) \end{aligned} \tag{7.57}$$

and exploiting the small-angle approximations where possible,

$$\sin(\alpha + \beta + \chi) \approx \sin(\alpha + \beta) + \cos(\alpha)\sin(\chi) \tag{7.58}$$

The contents of the upper ROM in Figure 7.34 then contains the quantity $\sin(\alpha + \beta)$ and the lower ROM contains the quantity $\cos(\alpha)\sin(\chi)$. Since $\sin(\chi) \ll 1$, the width of the lower ROM can normally be made considerably smaller than the upper ROM.

An alternative means for calculating the ROM values based on numerical optimization is given in [11] also. Rather than be restricted to strictly a trigonometric interpretation, this approach permits each table value to be optimized independently. In the example where $A = B = C = 4$ and $D = 12$, this approach produced worst case spurious performance of approximately -84.2 dBc in comparison to -72.2 dBc using the Sunderland technique. The additional 12-dB performance improvement afforded by this numerical optimization technique is noteworthy.

Other optimizations and enhancements in the $\sin(\theta)$ compression area are certainly possible. One particular technique exploits the first-order Taylor series expansion for $\sin(\theta)$ by storing the alternate function

$$f(x) = \sin\left(\frac{\pi}{2}x\right) - \frac{\pi}{2}x \tag{7.59}$$

rather than $\sin(\theta)$ explicitly. Owing to the smaller dynamic range required to store $f(x)$, storage requirements can be further reduced. The required table lookup modification for use with this method is shown in Figure 7.35.

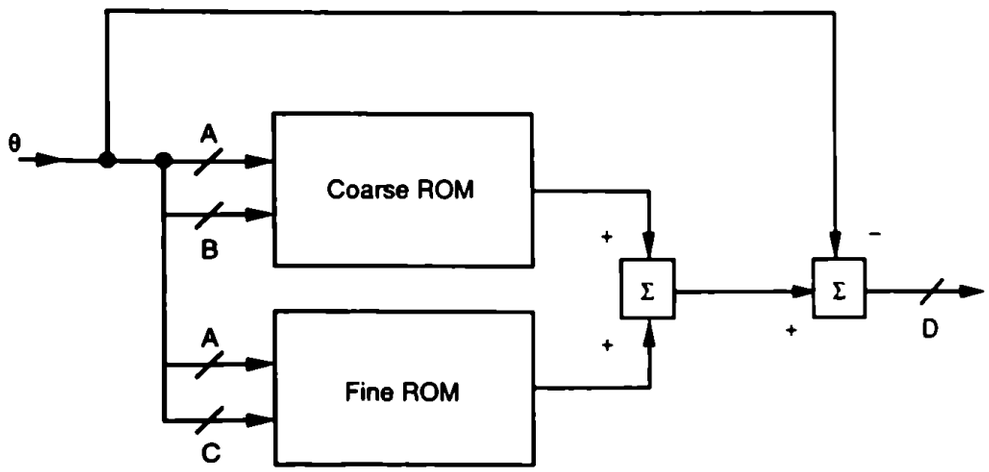

Figure 7.35 Modified architecture for $\sin(\theta)$ table compression using $f(\theta) = \sin(\theta) - \theta$. (© 1988 IEEE; reprinted with permission). After: [11], Figure 9.

A very interesting additional alternative for computation of $\sin(\theta)$ is discussed at length in [13] where combinatorial binary logic is used to approximate the sine function. The basic algorithm for trigonometric function approximation follows these guidelines:

1. Express the function or a related function as a multiplication by
   (a) expressing the binary operands as polynomials,
   (b) taking the derivative of the corresponding inverse trigonometric function with the operands expressed as polynomials. The resulting equation is a multiplication of polynomials.
2. Expand multiplication into a partial products array.
3. Prohibit carry propagation between array columns for the most significant redundant digits of the unknown operand.
4. Express the resulting equation in a partial product array.
5. Reduce the array using Boolean and algebraic equivalences.

In the case of $\sin(\theta)$, let Y equal the binary representation for $\sin(\theta)$ and θ be the binary representation for the angle argument. Then for $0 \leq Y \leq 1$,

$$Y = \sum_{i=0}^{N} y_i 2^{-i} = \sum_{i=0}^{N} y_i x^i \tag{7.60}$$

and for the angle θ where $0 \leq \theta \leq \pi/2$,

$$\theta = \sum_{i=0}^{N} \theta_i 2^{-i+1} = 2 \sum_{i=0}^{N} \theta_i x^i \tag{7.61}$$

The polynomial representation is used because it can be differentiated. Following step 1 of the algorithm, the necessary formula are obtained as

$$\begin{aligned} \theta(x) &= \sin^{-1}[Y(x)] \\ \frac{d\theta(x)}{dx} &= \frac{1}{\sqrt{1 - Y^2(x)}} \frac{dY(x)}{dx} \\ Y(x) &= \sqrt{1 - Y^2(x)}\, \theta'(x) \\ [Y'(x)]^2 &= [1 - Y^2(x)][\theta'(x)]^2 \end{aligned} \tag{7.62}$$

Following through the considerable amount of algebra as described in [13], the 12-bit estimate derived for $\sin(\theta)$ is as shown in Figure 7.36 where each column entry is a logic minterm and the overbars indicate complement. This figure completely describes the combinatorial logic required to estimate $\sin(\theta)$.

A comparison of this approach with more traditional methods is given in Figure 7.37. As shown, the 12-bit formulation is approximately equivalent to a third-order

Y1	Y2	Y3	Y4	Y5	Y6	Y7	Y8	Y9	Y10	Y11	Y12
1/2	1/4	1/8	1/16	1/32	1/64	1/132	1/256	1/512	1/1024	1/2048	1/4096
2	2	2	4	10	6	10	11	12	10	8	10 = 87
θ_1	$\theta_1\overline{\theta_2}$	$-\theta_1\theta_3$	$\theta_1\overline{\theta_2}$	$-\theta_2\theta_3$	$-\overline{\theta_1}\theta_2\overline{\theta_3}\overline{\theta_4}$	$-\overline{\theta_1}\theta_2\theta_4$	$-\theta_1\theta_2\theta_3\theta_4$	$-\theta_1\theta_2\theta_4$	$-\theta_2\overline{\theta_3}$	$-\overline{\theta_1}\theta_3$	$-\overline{\theta_1}\theta_3$
θ_2	θ_3	θ_4	$-\theta_1\theta_2\theta_3$	$-\theta_1\theta_2\theta_4$	$-\theta_1\theta_2\theta_5$	$-\overline{\theta_1}\theta_3\theta_4$	$\theta_1\theta_2\overline{\theta_3}\overline{\theta_5}$	$-\theta_3\overline{\theta_4}$	$-\theta_1\theta_3\theta_4$	$\theta_1\theta_4$	$\theta_2\theta_3$
			$-\theta_1\theta_4$	$-\theta_1\theta_3\theta_4$	$\theta_1\theta_3\overline{\theta_5}$	$\theta_2\overline{\theta_3}\theta_5$	$\theta_2\theta_4\overline{\theta_5}$	$-\theta_1\theta_2\theta_5$	$\theta_1\theta_4\theta_5$	$\theta_1\theta_5$	$-\overline{\theta_2}\theta_4$
			$\overline{\theta_2}\theta_5$	$\theta_1\overline{\theta_5}$	$\overline{\theta_1}\theta_7$	$-\theta_1\theta_4\theta_5$	$\overline{\theta_1}\theta_2\theta_6$	$-\overline{\theta_1}\theta_3\theta_5$	$\theta_1\theta_6$	$\theta_1\theta_2\overline{\theta_5}$	$\theta_1\theta_2\theta_4$
				$-\theta_1\theta_3$	$-\theta_2\overline{\theta_5}$	$-\overline{\theta_1}\theta_2\theta_6$	$-\theta_2\theta_3\theta_6$	$-\theta_3\theta_4\theta_5$	$-\overline{\theta_1}\theta_3\theta_6$	$\theta_1\theta_3\theta_5$	$\theta_1\theta_3\theta_5$
				$-\theta_2\theta_3$	$\theta_1\overline{\theta_2}\theta_6$	$-\theta_1\theta_3\theta_6$	$-\theta_1\theta_4\theta_6$	$-\theta_2\theta_4\theta_6$	$-\theta_1\overline{\theta_2}\theta_7$	$\theta_1\theta_2\theta_6$	$\theta_1\theta_4\theta_5$
				$-\theta_2\theta_4$		$\theta_1\overline{\theta_2}\theta_7$	$-\theta_1\theta_3\theta_7$	$-\theta_1\theta_5\theta_6$	$-\theta_2\theta_8$	θ_{12}	$-\theta_1\overline{\theta_2}\overline{\theta_6}$
				$\overline{\theta_1}\theta_6$		$\overline{\theta_1}\theta_8$	$\theta_1\overline{\theta_2}\theta_8$	$\theta_2\overline{\theta_3}\theta_7$	$-\theta_1\theta_{10}$	$\theta_1\theta_2\theta_6$	$\theta_1\theta_3\theta_6$
				$-\theta_2\overline{\theta_5}$		θ_2	$\overline{\theta_1}\theta_9$	$-\theta_1\overline{\theta_4}\overline{\theta_7}$	θ_{11}		$-\theta_1\overline{\theta_2}\theta_7$
				$\overline{\theta_1}\theta_2$		$\theta_1\theta_2\theta_3$	$\theta_2\overline{\theta_7}$	$-\theta_1\theta_3\theta_8$	$-\overline{\theta_1}\theta_2\theta_5$		$\theta_1\theta_8$
							$-\theta_1\theta_4$	$\theta_1\overline{\theta_2}\theta_9$			
								θ_{10}			

Figure 7.36 Reduced sine formulations for a 12-bit estimate. (© 1992 IEEE; reprinted with permission) After: [13]

(a)

Method	Bits or Order	Bits Correct Ave	Bits Correct Min.	Array Size Max Col.	Array Size Total Ele.	Latency Mpy	Latency Add
Prop.	8 b	7.14	3.78	5	26	< 1	0
Prop.	12 b	10.35	6.89	12	87	< 1	0
Prop.	16 b	13.52	9.49	33	250	< 1	0
Taylor	1 IRD	5.02	0.81	–	–	0	0
Taylor	3 ORD	10.88	3.73	–	–	3	1
Taylor	5 ORD	17.73	7.79	–	–	4	2
Cheby.	1 ORD	3.88	2.83	–	–	1	0
Cheby.	3 ORD	8.81	7.78	–	–	3	1
Cheby.	5 ORD	14.85	13.84	–	–	4	2

(b)

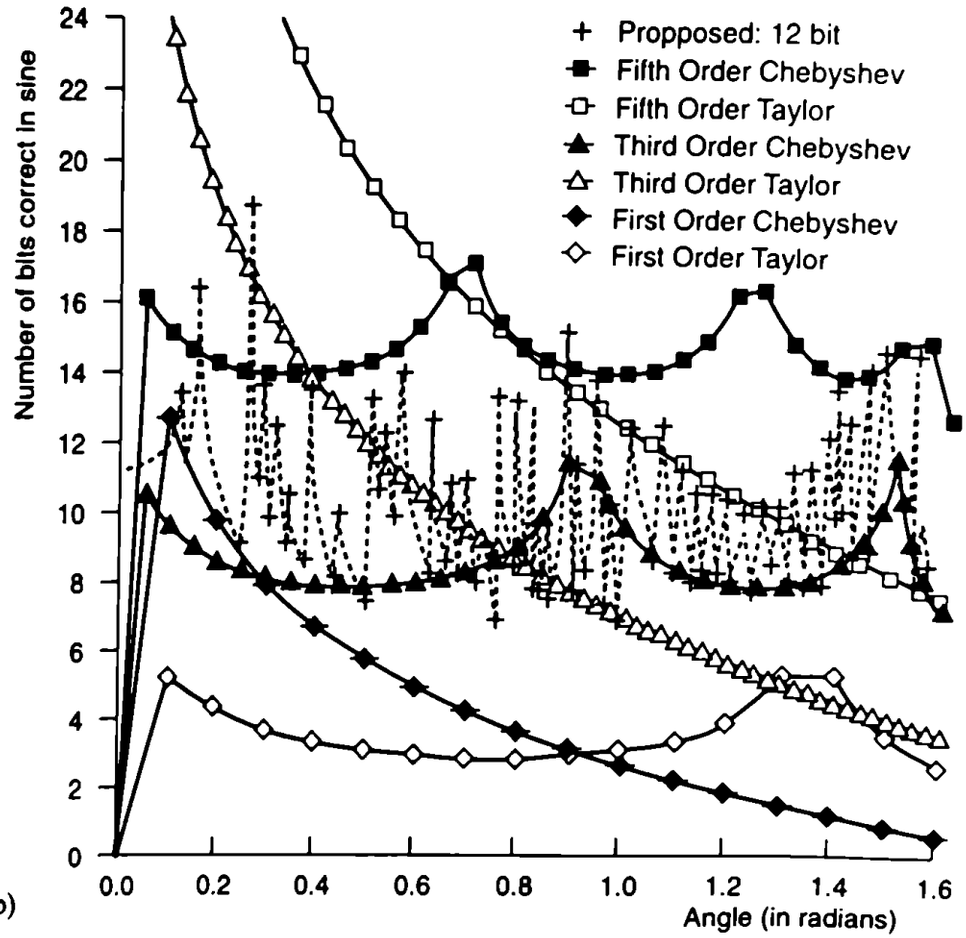

Figure 7.37 (a) Statistics of proposed, Taylor series, and Chebyshev polynomial methods for sine function. (b) Number of correct bits in proposed, Taylor, and Chebyshev methods for sine function. (© 1992 IEEE; reprinted with permission). Source: [13].

Chebyshev formulation. Although not directly comparable to the Sunderland approach described earlier, the underlying concepts appear very attractive for VLSI implementation.

7.3.6 Digital-to-Analog Converter Imperfections

The D/A converter is ultimately responsible for interfacing the digital world to the continuous RF/analog world. Although a number of performance measures have been standardized to quantify D/A converter imperfections, it is almost impossible to reflect these quantities to the DDS output and determine spectral purity. So, although these measures provide some evaluations guidelines for DACs, normally DDS spectral purity with a specific DAC must be evaluated on a case-by-case basis. A thorough introduction to this important aspect of DDS design can be found in [14–17].

7.3.7 Spurious Suppression in Direct Digital Synthesizers Using Numerical Techniques

Additional digital techniques may be incorporated into the generic DDS in order to largely eliminate the presence of discrete spurious signals at the DDS output. Normally, this results in a slight increase in wideband spectral noise but the impact can generally be made negligible.

From a standpoint of terminology, the earliest practitioners referred to these techniques as *dithering*[18] since randomization techniques are used to destroy the coherence of the undesired spurious components. Delta-sigma modulation techniques [19] based on digital signal processing techniques have spawned an alternative means for destroying these same discrete spectral components and are frequently referred to as *noise-shaping methods*. Unlike the randomization approaches that result in a white broadband output noise floor spectrum, these techniques normally produce close-in noise spectra, which are generally high-pass in nature in that phase noise very near the carrier is considerably better than at frequency offsets far removed from the carrier. Both of these methods are considered in this section.

A marriage between these two techniques is in principle possible as well. Rather than employ a randomization sequence in the dithering technique that is spectrally white, the random sequence could be processed by a digital high-pass filter prior to application in the dithering process. Although such a technique has to date not been observed in the literature, it is anticipated that performance similar to the noise-shaping approach could be achieved.

As of this writing, the author is only aware of one commercially available highly integrated DDS device that includes a means for discrete spurious suppression