An Efficient Pipeline Direct Digital Frequency Synthesizer Based on a Novel Interpolation Algorithm

Meng-Hsueh Lin*, Chun-Sheng Yang*, Ching-Yung Chu* and Sau-Gee Chen*

Abstract - **This paper proposes a new direct digital frequency synthesizer (DDFS). It has the merits of high speed, low complexity and high spectrum purity. It is based on a novel interpolation algorithm for sinusoidal functions. The algorithm accurately characterizes the interpolation error. With a small lookup table of $2^{N/4-1}$ words, the DDFS successively interpolate the target value in a pipeline fashion using only N addition operations, where N is the output word length. The DDFS is non-recursive, which is free of error-propagation problem. Simulation shows that for N=16-bit example, 100 dBc of SFDR (spurious free dynamic range) is achieved, with a lookup table of only 8 entries. In addition, due to its pipeline structure, the new DDFS have a very high throughput rate and a very short cycle time of about an adder delay.**

1 Introduction

A direct digital frequency synthesizer (DDFS) [1] is an efficient, programmable, powerful digital sine wave generator. It has high spectrum purity, wide programmable frequency range, fine frequency resolution, low phase noise and fast switching capability. As such, it is good for phase modulation, frequency modulation, spread-spectrum and frequency-hopping communication systems. Therefore, it is highly desirable to design a DDFS with high-speed, low complexity, wide frequency range and high accuracy.

A DDFS mainly consists of a phase accumulator and a sine/cosine, generator, as shown in Fig. 1 [3]. The phase accumulator's value specifies the instantaneous phase, which is fed as the argument to a sine/cosine generator that computes the digital sine and/or cosine values. The digital values are then converted to their analog values by a digital-to-analog converter (DAC), followed by a low-pass filtered (LPF) to remove all high-frequency noises. DDFS's can be categorized as recursive [2] and non-recursive types [3], [4]. The recursive DDFS's are notorious for their error-propagation problem, due to their error feedback nature. Moreover, due to recursion, they are hard to pipeline in implementation. In contrast, the non-recursive DDFS's do not have such problem. Regardless of recursive or non-recursive computations, there are various algorithms for DDFS design, including methods of table lookup [1], [5], [6], CORDIC algorithm [3], [7], [8], and polynomial approximation [4]. The table-lookup DDFS's have the trouble of large table size. Therefore, they are mostly feasible for short word lengths and hence have low SFDR's. There are techniques for reducing the table size, such as utilizing the symmetry of sinusoidal functions [6] and compressing the data stored in the table [9]. Polynomial-based DDFS's normally need multiplication operations and considerable table sizes. In comparison, the CORDIC-based DDFS's are highly efficient, due to their simplicity. Specifically, the non-recursive CORDIC-based pipeline DDFS [3] is regarded as a very good one. Still, the DDFS needs a ROM table of size $2^{N/3}$, meanwhile it has to generate both sine and cosine functions, even when only one of them is needed.

Here, we would like to propose a new efficient non-recursive DDFS with a smaller table size of $2^{N/4-1}$, while produces a higher SFDR, at a smaller complexity, as described below.

*Department of Electronics Engineering, National Chiao Tung University, 1001 Ta Hsueh Road, Hsinchu, Taiwan, ROC. E-mail: sgchen@cc.nctu.edu.tw, Tel: +886-3-5731625, Fax: +886-3-5710580.

2 The New Interpolation Algorithm for cos/sin Function

The following discussion is for cosine function. However, it also applies to sine function. Let's begin with the problem of middle-point interpolation. Given two known cosine values of $\cos(\theta+\Delta\theta)$ and $\cos(\theta-\Delta\theta)$, $\cos\theta$ can be solved from these two values as follows. In Taylor's series expansion:

$$\cos(\theta+\Delta\theta)=\cos\theta-(\Delta\theta)\sin\theta-\frac{1}{2}(\Delta\theta)^2\cos\theta+\frac{1}{3!}(\Delta\theta)^3\sin\theta+\frac{1}{4!}(\Delta\theta)^4\cos\theta-\frac{1}{5!}(\Delta\theta)^5\sin\theta+\cdots \quad (1)$$

and

$$\cos(\theta-\Delta\theta)=\cos\theta-(-\Delta\theta)\sin\theta-\frac{1}{2}(-\Delta\theta)^2\cos\theta+\frac{1}{3!}(-\Delta\theta)^3\sin\theta+\frac{1}{4!}(-\Delta\theta)^4\cos\theta-\frac{1}{5!}(-\Delta\theta)^5\sin\theta+\cdots \quad (2)$$

Summing up these two equations, we have

$$\begin{aligned}[\cos(\theta+\Delta\theta)+\cos(\theta-\Delta\theta)]/2&=\cos\theta-\frac{1}{2}(-\Delta\theta)^2\cos\theta+\frac{1}{4!}(-\Delta\theta)^4\cos\theta-\frac{1}{6!}(-\Delta\theta)^6\cos\theta+\cdots \quad (3)\\&=\cos\theta[1-\frac{1}{2}(\Delta\theta)^2+\frac{1}{4!}(\Delta\theta)^4-\frac{1}{6!}(\Delta\theta)^6+\cdots]\end{aligned}$$

As a result, we can analytically obtain the interpolation value $\cos\theta$ as:

$$\cos\theta = [1+\frac{1}{2}(\Delta\theta)^2+\frac{5}{24}(\Delta\theta)^4+\cdots]\times [\cos(\theta+\Delta\theta)+\cos(\theta-\Delta\theta)]/2 \quad (4)$$

Similarly for sine function,

$$\sin\theta = [1+\frac{1}{2}(\Delta\theta)^2+\frac{5}{24}(\Delta\theta)^4+\cdots]\times [\sin(\theta+\Delta\theta)+\sin(\theta-\Delta\theta)]/2 \quad (5)$$

In realizing these equations with *N*-bit accuracy, we can have the following three feasible algorithms, depending on the magnitude of $\Delta\theta$.

2.1 The 0th-order interpolation algorithm

With *N*-bit fractional accuracy, if $\Delta\theta \le 2^{-N/2}$, the interpolation equation reduces to

$$\cos\theta = [\cos(\theta+\Delta\theta)+\cos(\theta-\Delta\theta)]/2 \quad (6)$$

which correspond to an error ε of

$$\varepsilon = [\frac{1}{2}(\Delta\theta)^2+\frac{5}{24}(\Delta\theta)^4+\cdots][\frac{\cos(\theta+\Delta\theta)+\cos(\theta-\Delta\theta)}{2}] \quad (7)$$

It can be shown that $|\varepsilon| < 2^{-N}$. This algorithm requires a lookup table of size $2^{N/2-1}$, without multiplication operation. Practical realization of the algorithm is as follows.

Given $\theta = 0.a_1a_2\cdots a_N$ and $\theta_i \equiv 0.a_1a_2\cdots a_i$ in binary representation, we first retrieve $\cos(\theta_{N/2-1}+2^{-(N/2-1)})$ and $\cos\theta_{N/2-1}$ by addressing the *N*/2-1 MSB's of θ to a lookup table (of size $2^{N/2-1}$), then solve $\cos(\theta_{N/2-1}+2^{-N/2})$ by simply averaging these two values. For the final $\cos\theta$ value, we need to trace all remaining bits from $a_{N/2}$ to a_{N-1}, and successively solve all the *N*/2 intermediate confining cosine values from $\cos(\theta_{N/2-1}+2^{-N/2})$ to $\cos(\theta_{N-1}+2^{-N})$, using the add-and-divide-by-two operations. As a result, for the final $\cos\theta = \cos\theta_N$ value, we need *N*/2 additions.

2.2 The 2nd-order interpolation algorithm

If $\Delta\theta \le 2^{-N/4}$, then eq. (4) reduces to

$$\cos\theta = [1+\frac{1}{2}(\Delta\theta)^2][\cos(\theta+\Delta\theta)+\cos(\theta-\Delta\theta)]/2 \quad (8)$$

which correspond to an error ε of

$$\varepsilon = [\frac{5}{24}(\Delta\theta)^4+\cdots][\cos(\theta+\Delta\theta)+\cos(\theta-\Delta\theta)]/2 \quad (9)$$

It can be shown that $|\varepsilon| < 2^{-(N+2)}$. This algorithm requires a lookup table of size $2^{N/4-1}$, and seemingly needs two multiplication operations. However as will be discussed subsequently, for the computation of the final $\cos\theta$, it only needs *N* addition operations. Similar to the 0th-order algorithm, it begins with the same action of retrieving from a lookup table (of size $2^{N/4-1}$) and averaging $\cos(\theta_{N/4-1}+2^{-(N/4-1)})$ and $\cos\theta_{N/4-1}$. The average value is then multiplied by the corresponding factor $1+\frac{1}{2}(\Delta\theta_{N/4})^2 = 1+\frac{1}{2}(2^{-N/4})^2 = 1+2^{-(N/2+1)}$. Note that this operation only needs one addition operation, instead of two multiplications as mentioned before. Proceed in a similar manner from bits $a_{N/4}$ to a_N, we successively solve all the intermediate confining cosine values from $\cos(\theta_{N/4-1}+2^{-N/4})$ to $\cos(\theta_N+2^{-N})$, all involving the averaging operation followed by the multiplication by the term $1+\frac{1}{2}(\Delta\theta_m)^2 = 1+\frac{1}{2}(2^{-m})^2 = 1+2^{-(2m+1)}$. In all, for each intermediate cosine value, two addition operations are required. However, when $m \ge N/2$, $1+\frac{1}{2}(2^{-m})^2$ reduces to 1 for *N*-bit accuracy. As such, for the final $\cos\theta$ value, *N* addition operations are enough.

2.3 The 4th-order interpolation algorithm

If $\Delta\theta \le 2^{-N/6}$, eq. (4) reduces to

$$\sin\theta = [1+\frac{1}{2}(\Delta\theta)^2+\frac{5}{24}(\Delta\theta)^4]\times [\sin(\theta+\Delta\theta)+\sin(\theta-\Delta\theta)]/2 \quad (10)$$

which correspond to an error ε of

$$\varepsilon = [\frac{7}{360}(\Delta\theta)^6+\cdots][\cos(\theta+\Delta\theta)+\cos(\theta-\Delta\theta)]/2 \quad (11)$$

It can be shown that $|\varepsilon| < 2^{-(N+5)}$. This algorithm requires a lookup table of size $2^{N/6-1}$, and seemingly needs four multiplication operations. However, similar to the 2nd-order algorithm they can be reduced to a much simpler one, whose detail will be reported in the final paper.

To clarify and simplify the illustration of the proposed pipeline DDFS, we only consider the design of the new pipeline DDFS based on the 2nd-order interpolation algorithm. We regard it as more efficient than the other two alternatives. Similar structures can be easily obtained for the 0th-order and 4th-order algorithms.

3 The New Pipeline DDFS

As shown in Fig. 2 is the new pipeline DDFS for cosine function, based on the 2nd-order interpolation algorithm (i.e., eq. (8)). Sine function can be computed similarly. Let $\theta = \times\times\times.a_1a_2\cdots a_N$, be mapped from a binary input $B = f\cdot t = \times\times\times.b_1b_2\cdots b_N$, where *f* is the desired output frequency and *t* is the time parameter. Note that the three-bit integer part is for covering full 2π phase range, and. $B = 001.00\cdots0$ maps to $\theta = \pi/4$. Here, we assume *N*=16. Structure description and operation of the new pipeline DDFS (of Fig. (2)) is detailed below.

Stage1: The Accumulator block is for the incremental generation (at a step of $F_{clk}\alpha/2^M$) of the input binary $B = f \cdot t$ values to the DDFS , where M=36 is assumed here, which controls the resolution of the output frequency, F_{clk} is the frequency of the system clock, "α" is the frequency control word, and B is composed of 3 integer bits and 33 fractional bits..

Stage2: The 1st Quadrant Mirror block performs symmetry reduction of the input phase to $\pi/4$, for reducing table size [3]. This stage strips off the integer part of the input phase. In Fig. 2, only 22 MSB's of the accumulator's output is retained for lower complexity consideration.

Stage3: The $\pi/4$ Multiplier block [3] maps its fractional input to the phase range of $[0,\pi/4]$, using the equation $\theta = B \cdot \pi/4 = 0.a_1a_2 \cdots a_N$. By a a π /4 as $2^{-1} + 2^{-2} + 2^{-5} + 2^{-8} + 2^{12}$, the multiplier block can be realized by a few addition operations.

Stage4: The Sin/Cos Generator block realizes the 2nd-order interpolation algorithm detailed in the previous section. It consists of a lookup ROM table, A*i* and B*i* blocks, as will be detailed next.

Stage5: The Mapping block is for the final sign correction of the results.

ROM Table: This block stores all the eight (i.e., $2^{N/4-1}$) cosine values addressed by the three fractional MSB bits $0.a_1a_2a_3$ of θ, for the retrieval of $\cos(0.a_1a_2a_3 + 0.001)$ and $\cos(0.a_1a_2a_3)$.

A1 stage: The A1 block solves $\cos(0.a_1a_2a_31)$, using

$$[1+\frac{1}{2}(\Delta\theta_4)^2][\cos(0.a_1a_2a_3)+\cos(0.a_1a_2a_3+0.001)]/2$$
$$=(1+2^{-9})[\cos(0.a_1a_2a_3)+\cos(0.a_1a_2a_3+0.001)]/2$$

Note that $\Delta\theta_4 = 2^{-4}$ (i.e., $\Delta\theta_{N/4} = 2^{-N/4}$). This stage requires two additions.

A2 stage: The A2 block checks bit a_4 to decide if θ lies in the phase ranges of $a_1a_2a_3$ $a_1a_2a_3$) or $[0.a_1a_2a_31, 0.a_1a_2a_3 + 0.001)$. Then, similar interpolation operation as in A1 stage is done to find $\cos(0.a_1a_2a_3a_41)$, where $\Delta\theta_5 = 2^{-5}$. So do the pipeline stages A3 to $A_{N/4}$=A_4. Detailed architecture of these stages are shown in Fig. 4.

B1 to B9 stages (i.e., 1 to N/2+1=9):

These pipeline stages have similar operations to that of A*i* stages, except that here $(\Delta\theta_i)^2/2 \le 2^{-(N+1)} = 2^{-17}$ can be omitted. Hence, the interpolation reduces to a single addition. The B9 stage is the output stage. Detailed architecture of these stages are shown in Fig. 5.

As discussed before, in total it requires N additions to compute a cosine value. The final results are then sent to a digital-to-analog converter, followed by an analog low-pass filter for waveform smoothing.

4 Performance Evaluation

Table 1 summarizes performance figures of the new DDFS's due to equations (6) and (8), and the highly efficient pipeline CORDIC-based DDFS of [3]. For short word length such as smaller than 8 bits, the DDFS based on eq. (6) may be a better option than the other two. For higher accuracy such as 16 bits, DDFS based on eq. (8) is a better choice. Note that even when only one of the sine and cosine functions needs to be computed, the DDFS of [3] always has to calculate both of them. As such, DDFS of [3] is efficient when both functions are required. DDFS due to eq. (8) is generally a better choice over the other two for the computation of a single sinusoidal function. Simulations show that SFDR of 100 dBc can be achieved, as depicted in Fig. 6, where the vertical scale and horizontal scale correspond to dB value and frequency, respectively. The tallest spur in the figure corresponds to the desired simulated output frequency, while the other spurs represent the spectrum impurity due to finite-precision error.

5 Conclusion

The new DDFS based on eq. (8) has the merits of high speed, high spectrum purity, low complexity, and small area. We can further reduce the table size to $2^{N/6-1}$ by including the 4th-order term as in eq. (10), at the expense of a few more additions. Of course one can combine those popular ROM size reduction techniques as in [6], and [9] to further reduce table size. All these are currently under investigation, including VLSI hardware simulation and realization of the new DDFS, which will be reported in the final complete version.

References

[1] J. Tierney, C. M. Rader, and B. Gold, "A digital frequency synthesizer," *IEEE Trans. Audio Electroacoust.,* vol. AU-19, pp. 48–56, Mar. 1971.

[2] N. J. Fliege, "Complex digital oscillators and FSK modulators," *IEEE Trans. Signal Processing,* vol. 40, pp. 333–342, Feb. 1992.

[3] 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 Journal of Solid-State Circuits*, vol. 34, no. 8, pp. 1034-1043, Aug. 1999.

[4] 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, pp. 385-390, March 2000.

[5] H. T. Nicholas and H. Samueli, "A 150 MHz direct digital frequency synthesizer in 1.25-µm CMOS with 90 dBc spurious performance," *IEEE J. Solid-State Circuits,* vol. 26, pp. 1959–1969, Dec. 1991.

[6] L. K. Tan and H. Samueli, "A 200 MHz quadrature digital synthesizer/mixer in 0.8µm CMOS," *IEEE J. Solid-State Circuits*, vol. 30, pp. 193–200, Mar. 1995.

[7] J. Volder, "The CORDIC trigonometric computing technique," *IEEE Trans. Computers,* vol. EC-8, pp. 330–334, Sept. 1959.

[8] G. Gielis, R. van de Plassche, and J. van Valburg, "A 540 MHz 10b polar to Cartesian converter," *IEEE J. Solid-State Circuits,* vol. 26, pp. 1645–1650, Nov. 1991.

[9] B. H. Hutchison, Jr., *Frequency Synthesis and Applications*, New York: IEEE Press, 1975.

Table 1. Performance comparison of DDFS's

	ROM size	No. of adders
New DDFS using eq. (8)	$2^{N/4-1}$	N
New DDFS using eq. (6)	$2^{N/2-1}$	$N/2$
DDFS of Madisetti, et al., [3]	$2^{N/3}$	$3N/2$

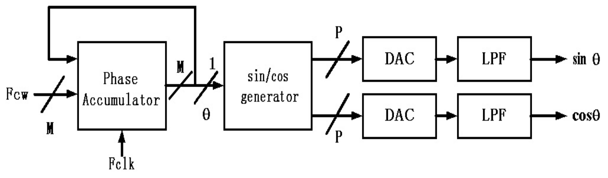


Fig. 1. A conventional DDFS structure.

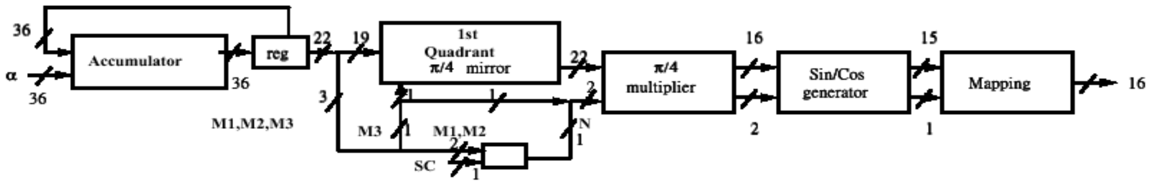


Fig. 2. Structure of the whole DDFS.

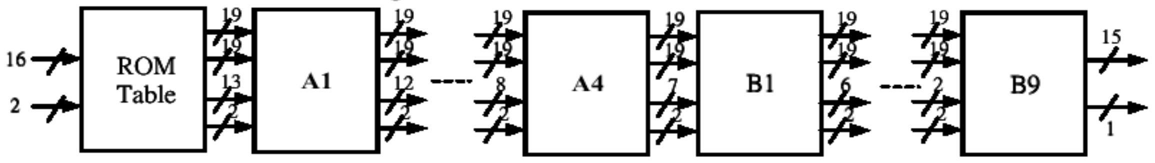
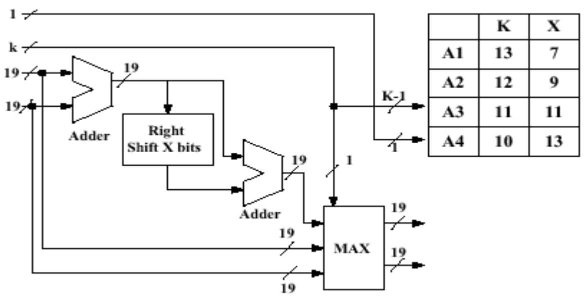
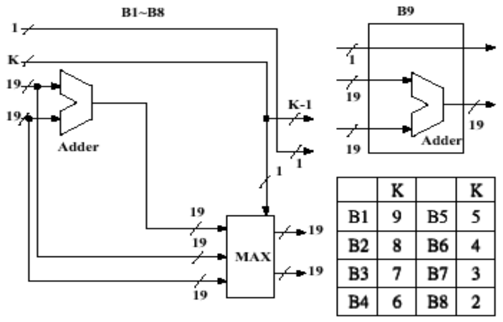


Fig. 3. Detailed pipeline structure of the Sin/Cos Generator block of the new DDFS.



	K	X
A1	13	7
A2	12	9
A3	11	11
A4	10	13

Fig. 4. The type-A pipelined stage



	K		K
B1	9	B5	5
B2	8	B6	4
B3	7	B7	3
B4	6	B8	2

Fig. 5. The type-B pipelined stage

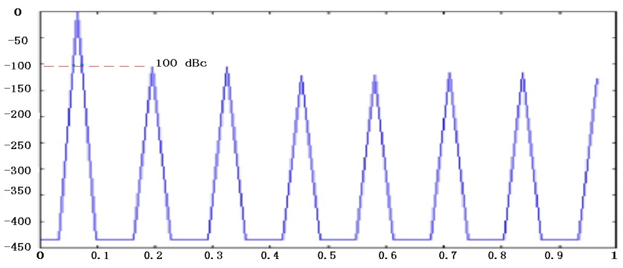


Fig. 6. Spectrum and SDFR plot of the new DDFS