Diophantine Frequency Synthesis

Paul Peter Sotiriadis, *Member, IEEE*

***Abstract*—A methodology for fine-step, fast-hopping, low-spurs phase-locked loop based frequency synthesis is presented. It uses mathematical properties of integer numbers and linear Diophantine equations to overcome the constraining relation between frequency step and phase-comparator frequency that is inherent in conventional phase-locked loop based frequency synthesis. The methodology leads to fine-step, fast-hopping, modular-structured frequency synthesizers with potentially very low spurs, especially in the vicinity of the carrier. The paper focuses on the mathematical principles of the new methodology and the related number theoretic algorithms.**

I. Introduction

Fine-frequency synthesis[1] is fundamentally important to positioning and navigation (GPS), time keeping (atomic clocks), scientific instrumentation, certain radar and communication systems, and many other applications.

Several fine-step frequency synthesis architectures have been proposed; a rich collection can be found in [1] and [2]. The different schemes can be roughly organized into three classes: multi-loop, fractional, and direct digital synthesizers. Many hybrids also exist.

Multi-loop architectures [1] can provide clean output signal at the cost of complexity, physical size, and, in certain cases, of frequency hopping speed. Complex hardware implementations are typical and large dividers are not uncommon.

Fractional synthesizers [2] provide fine frequency resolution and fast hopping with low complexity hardware, but they suffer from spurious signals very close to the carrier due to their inherent weak FM modulation.

Direct digital synthesis (DDS) [3] (or numerical oscillators) is a convenient approach to fine step, large range, and fast hopping frequency synthesis using standardized building blocks. In almost all cases, the output signal is not purely periodic because of truncation errors generated in the phase accumulator and the digital-to-analog converter. This results in spurious signals very close to the carrier [4]–[7]. The general spectral purity is also limited by the digital-to-analog converter [3], [8]. Also, DDS usually results in higher power consumption than other approaches (e.g., fractional-N synthesizers).

This paper introduces the Diophantine Frequency Synthesis[2] (DFS), a new approach to fine-step frequency synthesis (e.g., 10^{-10} fractional resolution) that is based on mathematical properties of integer numbers and linear Diophantine equations. By definition, Diophantine equations are algebraic equations whose solutions are required to be integers [9].

DFS uses two or more basic phase-locked loops (PLLs). The output frequencies of the PLLs are added or subtracted to give the output frequency of the synthesizer. DFS provides the mathematical algorithm for choosing the fixed sizes of the prescalers and for adjusting the sizes of the feedback dividers.

DFS offers a significant advantage: it leads to PLL-based architectures for which the output frequency step can be made arbitrarily small (e.g., 10^{-10} fractional resolution) without using large prescalers or small reference frequencies. This allows for simultaneously having very small output frequency step and high phase-comparator frequencies resulting in large loop bandwidths and therefore fast frequency hopping. DFS *distributes* the frequency resolution among the PLLs.

This paper focuses on the mathematical foundations of DFS. A complete mathematical framework is introduced along with the algorithms needed to calculate all the required parameters of the synthesizers.

The paper is organized as follows: Section II introduces the notation and assumptions used throughout the paper. Section III introduces DFS through two simple examples by emphasizing intuition and avoiding mathematical details. Section IV discusses the general high-level architecture for DFS. Section V lays out the formal mathematical framework of DFS. Sections VI and VII present examples of fixed and variable frequency DFS schemes. Appendices A and B provide an additional lemma and a MATLAB implementation of the algorithm in Section V.

II. Notation and Assumptions

In this work we consider frequency synthesis architectures involving two or more basic PPLs[3] like the one in

Manuscript received January 25, 2006; accepted April 16, 2006. This work was supported in part by the Defense Technical Information Center 06MISP7 - Johns Hopkins University Applied Physics Laboratory, Laurel, MD.

The author is with the Department of Electrical and Computer Engineering, The Johns Hopkins University, Baltimore, MD 21218 (e-mail: pps@jhu.edu).

Digital Object Identifier 10.1109/TUFFC.2006.139

[1]The terms "fine (frequency) step," "high resolution," and "fine-frequency synthesis" are used interchangeably in this paper.

[2]Patent pending, Johns Hopkins University Applied Physics Lab.

[3]The steady state behavior of the basic PLL is the following: A periodic signal of frequency f_{in}, enters the prescaler (divider N) producing another periodic signal of frequency f_{in}/N at the one input of the phase comparator (PC). Similarly, the frequency f_{out} of the output periodic signal is divided by the feedback divider ($\hat{n}$) resulting in a signal of frequency $f_{\text{out}}/\hat{n}$ entering the other input of the

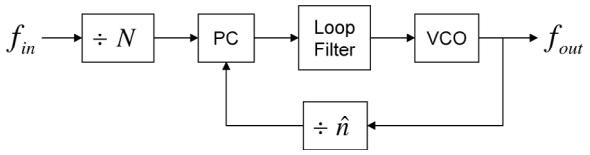

Fig. 1. Basic PLL.

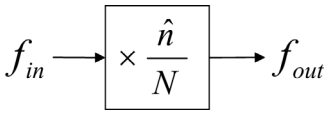

Fig. 2. Simplified schematic of the basic PLL.

Fig. 1. Note that the phase comparator (PC) may be a phase-frequency comparator as well.

Throughout this paper, the prescaler (divider N) is assumed to have a fixed size, N. The size of the feedback divider, $\hat{n}$, is the sum $\hat{n} = \bar{n} + n$, of a fixed value $\bar{n}$ and a variable n which can take both negative and positive values within a predefined range. For all values of n, $\hat{n}$ is positive. The output frequency of the PLL is

$$f_{\text{out}} = \frac{\hat{n}}{N} f_{\text{in}}.$$

Since the focus of this work is on high-level architecture of frequency synthesizers using basic PLLs and not in the technical details of the individual PLLs, Fig. 2 is used for convenience instead of Fig. 1. It is agreed, however, that simplification of the fraction $\hat{n}/N$ is *not* allowed, i.e., $\hat{n}/N$ and $k\hat{n}/(kN)$ correspond to two different PLLs.

Note that frequency synthesis using a single PLL, as that in Fig. 1, implies frequency steps that are equal to the phase-comparator frequency, i.e., equal to f_{in}/N. This means that small frequency step (using large N or/and small f_{in}) requires low phase-comparator frequency f_{in}/N and therefore small loop bandwidth [1], [2]. The last one implies slow frequency hopping and possibly increased spurious signals close to f_{out}. DFS overcomes these problems and allows for both high phase-comparator frequency and very small frequency step at the same time.

Mixing of two signals at frequencies f_1 and f_2 is denoted as in Fig. 3 where the outcome can be either $f_1 + f_2$ or $f_1 - f_2$. The context in the paper always indicates whether the sum or the difference is considered. Note however that, as shown in the following sections, the choice of the sum or the difference dictates only the central frequency of the output signal but neither the resolution nor the range of the DFS synthesizer.

Similar notation is used for the mixing of three or more signals. In a circuit implementation this can be done by sequentially mixing pairs of signals and the choice of the pairing usually influences the quality of the output.

phase comparator. The phase comparator derives the phase difference of the two signals and feeds it to the voltage controlled oscillator (VCO) through the loop filter. At the steady state, $f_{\text{out}}/\hat{n} = f_{\text{in}}/N$, any fluctuation of the phase difference $\Delta\phi$ between these two signals results in a correction of the VCO phase so that $\Delta\phi$ remains close to a predefined value. More information on PLLs can be found, for example, in [1], [2], and [10].

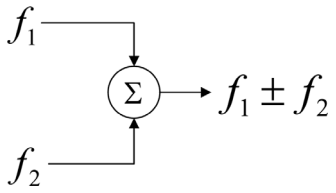

Fig. 3. Mixer.

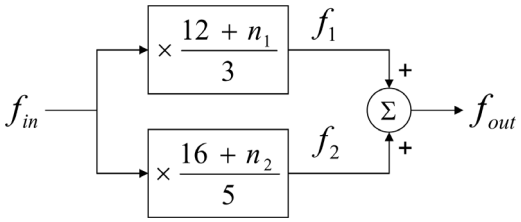

Fig. 4. A simple DFS scheme.

Minimization of the mixing spurs also involves the choice of the central frequencies and frequency ranges of the mixed signals, the choice of the sum or difference of their frequencies, and, of course, the type of the mixer. The examples in the paper provide some indications regarding these decisions; however, the goal of this paper is only to lay out the mathematical foundations of DFS.

III. Two Motivating Examples

Two examples of DFS synthesizers are discussed first to illustrate the intuition of the approach. The mathematical details are postponed to Section V.

A. Example 1

DFS uses mathematical properties of integer numbers to achieve a very fine frequency step without using large frequency dividers. Specifically, it combines the outputs of two or more PLLs with (small) prescalers, say, $N_1, N_2, \ldots, N_k$ (and small feedback dividers), to achieve output frequency resolution equal to $f_{\text{in}}/(N_1 N_2 \cdots N_k)$ and output frequency range $2f_{\text{in}}$. Small prescalers imply high phase-comparator frequencies, f_{in}/N_i, and fast frequency hopping.

Consider the simple architecture of Fig. 4 consisting of two PLLs whose output frequencies are summed giving

$$f_{\text{out}} = \left(\frac{n_1}{3} + \frac{n_2}{5} + \frac{12}{3} + \frac{16}{5}\right) f_{\text{in}}. \tag{1}$$

The prescalers are 3 and 5 (fixed). Let the feedback dividers, $12 + n_1$ and $16 + n_2$, be variable with

$$-3 \le n_1 \le 3 \quad \text{and} \quad -5 \le n_2 \le 5,$$

TABLE I
FREQUENCIES SYNTHESIZED BY THE DFS ARCHITECTURE OF FIG. 4.

$f_{\text{out}} = \left(\frac{n_1}{3} + \frac{n_2}{5} + \frac{12}{3} + \frac{16}{5}\right) f_{\text{in}}$						
n_1	n_2	$\frac{f_{\text{out}}}{f_{\text{in}}}$	=	$\left(\frac{n_1}{3} + \frac{n_2}{5}\right)$	+	$\left(\frac{12}{3} + \frac{16}{5}\right)$
−3	0	93/15	=	−15/15	+	108/15
−1	−3	94/15	=	−14/15	+	108/15
−2	−1	95/15	=	−13/15	+	108/15
−3	1	96/15	=	−12/15	+	108/15
−1	−2	97/15	=	−11/15	+	108/15
−2	0	98/15	=	−10/15	+	108/15
−3	2	99/15	=	−9/15	+	108/15
−1	−1	100/15	=	−8/15	+	108/15
−2	1	101/15	=	−7/15	+	108/15
−3	3	102/15	=	−6/15	+	108/15
−1	0	103/15	=	−5/15	+	108/15
−2	2	104/15	=	−4/15	+	108/15
−3	4	105/15	=	−3/15	+	108/15
−1	1	106/15	=	−2/15	+	108/15
−2	3	107/15	=	−1/15	+	108/15
0	0	108/15	=	0/15	+	108/15
−1	2	109/15	=	1/15	+	108/15
−2	4	110/15	=	2/15	+	108/15
0	1	111/15	=	3/15	+	108/15
−1	3	112/15	=	4/15	+	108/15
1	0	113/15	=	5/15	+	108/15
0	2	114/15	=	6/15	+	108/15
−1	4	115/15	=	7/15	+	108/15
1	1	116/15	=	8/15	+	108/15
0	3	117/15	=	9/15	+	108/15
2	0	118/15	=	10/15	+	108/15
1	2	119/15	=	11/15	+	108/15
0	4	120/15	=	12/15	+	108/15
2	1	121/15	=	13/15	+	108/15
1	3	122/15	=	14/15	+	108/15
3	0	123/15	=	15/15	+	108/15

i.e., the range of each feedback divider is twice the size of the corresponding prescaler (this specifies the required frequency range of the VCOs). Then, f_1 can take any of the 7 values:

$$f_1 \in \left\{ \frac{9}{3} f_{\text{in}},\ \frac{10}{3} f_{\text{in}},\ \frac{11}{3} f_{\text{in}}, \ldots, \frac{15}{3} f_{\text{in}} \right\}, \tag{2}$$

and f_2 can take any of the 11 values:

$$f_2 \in \left\{ \frac{11}{5} f_{\text{in}},\ \frac{12}{5} f_{\text{in}},\ \frac{13}{5} f_{\text{in}}, \ldots, \frac{21}{5} f_{\text{in}} \right\}. \tag{3}$$

Table I shows a set of output frequencies f_{out} that can be synthesized by appropriately choosing n_1 and n_2 within their specified ranges.

The mathematical principles behind Table I are discussed in detail for the general case in Section V. From a qualitative perspective, Table I demonstrates three properties of the simple DFS architecture in Fig. 4.

Property 1: The frequency step of the DFS scheme is constant and equals

$$\text{frequency step} = \frac{f_{\text{in}}}{15} = \frac{f_{\text{in}}}{3 \cdot 5}. \tag{4}$$

In other words, the particular choice of prescalers results in much smaller frequency step than those of the individual PLLs, i.e., $f_{\text{in}}/3$ and $f_{\text{in}}/5$, respectively.

Property 2: The output frequency range is $2f_{\text{in}}$. More accurately, by defining $\overline{f_{\text{out}}} = f_{\text{out}}|_{n_1=n_2=0}$, we have

$$f_{\text{out}} = \overline{f_{\text{out}}} - f_{\text{in}} \ \ldots \ \overline{f_{\text{out}}} + f_{\text{in}}. \tag{5}$$

Property 3: If the mixer provided the difference frequency between f_1 and f_2, instead of their sum, i.e., if $f_{\text{out}} = f_1 - f_2$, properties 1 and 2 would still hold. This is true because we assumed that n_1 and n_2 take values within the ranges $-N_1, \ldots, N_1$ and $-N_2, \ldots, N_2$, respectively, which are symmetric with respect to 0.

Note that properties 1, 2, and 3 result solely from the sum $\frac{n_1}{3} + \frac{n_2}{5}$ (or difference $\frac{n_1}{3} - \frac{n_2}{5}$) and are independent of the specific values of the constants $\bar{n}_1$ and $\bar{n}_2$.

Table I indicates that f_{out} can be expressed in the form

$$f_{\text{out}} = \overline{f_{\text{out}}} + \frac{a}{15} f_{\text{in}},$$

where a takes the values -15, -14, $-13, \ldots, 14$, 15, and the central frequency is $\overline{f_{\text{out}}} = \frac{108}{15} f_{\text{in}}$. In contrast, the output frequencies of the individual PLLs are

$$f_1 = \frac{12}{3} f_{\text{in}} + \frac{n_1}{3} f_{\text{in}}$$

and

$$f_2 = \frac{16}{5} f_{\text{in}} + \frac{n_2}{5} f_{\text{in}}.$$

Table I shows how to pick a pair of values (n_1, n_2) resulting in a specific value of a, i.e., how to solve the Diophantine equation

$$\frac{n_1}{3} + \frac{n_2}{5} = \frac{a}{15}$$

The relation between n_1, n_2, and a is nontrivial, and in some cases it is not unique even with the constraints $-3 \le n_1 \le 3$ and $-5 \le n_2 \le 5$; i.e., certain values of a result from more that one pair (n_1, n_2). For example, $a = -1$ results from both $(n_1, n_2) = (-2, 3)$ and $(n_1, n_2) = (1, -2)$.

Fig. 5 shows graphically the particular relation between the pair (n_1, n_2) and parameter a that was used to construct Table I.

Now let's consider the simple architecture in Fig. 4 but with prescalers 6 and 15 instead of 3 and 5, respectively. Now, n_1 ranges from -6 to 6 and n_2 ranges from -15 to 15. In the line of the previous example, one might expect that f_{out} would range from $\overline{f_{\text{out}}} - f_{\text{in}}$ to $\overline{f_{\text{out}}} + f_{\text{in}}$ and the frequency step would be $f_{\text{in}}/(6 \cdot 15) = f_{\text{in}}/90$.

By calculating the output frequencies corresponding to all allowed pairs (n_1, n_2) we see that the expected output range is indeed achievable; however, the frequency step (resolution) is *only* $f_{\text{in}}/30$ and therefore three times larger that the "expected" $f_{\text{in}}/(6 \cdot 15)$. Note that $\text{lcm}(6, 15) = 30 = 3 \cdot 5$, where lcm is the least common multiple function.

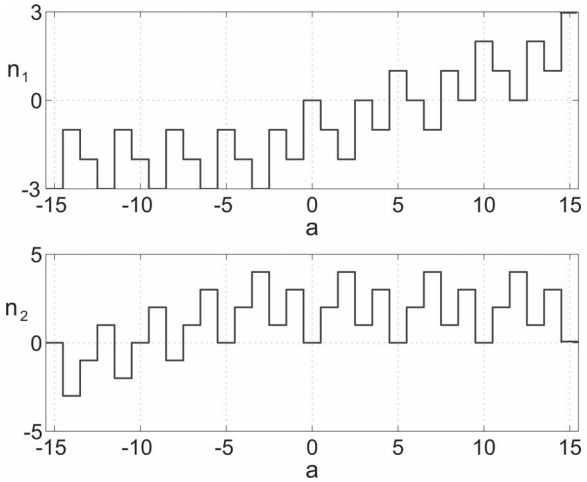

Fig. 5. Graphical representation of the relation between n_1, n_2, and a that was used to generate Table I.

Question: *What are the special qualities of the pair of numbers "3" and "5" leading to properties 1, 2, and 3?*

Answer: *They are pairwise relatively prime integers*, i.e., $\gcd(3,5)=1$ (where gcd is the greatest common divisor). This is not true for the pair $(6,15)$.

The formal answer to the above question is given in Section V. Note also that for both prescalers pairs (N_1, N_2), the frequency step is $f_{\text{in}}/\text{lcm}(N_1, N_2)$.

The choice of the constants $\bar{n}_1$ and $\bar{n}_2$ ($\bar{n}_1 = 12$ and $\bar{n}_2 = 16$ in Fig. 4) as well as the choice of $+$ or $-$ in the mixer define the central frequency $\overline{f_{\text{out}}}$ and the frequency ratio f_1/f_2 in the mixer. Therefore, these choices can be used to minimize the spurious signals generated by the mixer. Moreover, $\bar{n}_1$ and $\bar{n}_2$, along with N_1 and N_2, specify the required frequency ranges of the VCOs.

Remark: Henceforth, we concentrate only on the variable part, $\frac{n_1}{N_1} + \frac{n_2}{N_2}$, of the output frequency expression (1), which realizes the principle of DFS. In the general case of k PLLs, the corresponding expression is

$$\frac{n_1}{N_1} + \frac{n_2}{N_2} + \cdots + \frac{n_k}{N_k}, \tag{6}$$

where we always assume that $-N_i \le n_i \le N_i$ for all indices i.

Again, replacement of $\frac{n_i}{N_i}$ by $-\frac{n_i}{N_i}$ for any index i, i.e., substraction instead of addition in the mixing of the i^{th} signal, does not influence the output frequency range or the frequency step.

B. Example 2

A set of pairwise relatively prime[4] prescalers can be used to achieve extremely small frequency steps. Since the prescalers can be small in size, both tiny frequency steps *and* high phase-comparator frequencies at the PLLs can be achieved at the same time.

[4]The integers $N_1, N_2, \ldots, N_k$ are called pairwise relatively prime if $\gcd(N_i, N_j) = 1$ for all $i \neq j$.

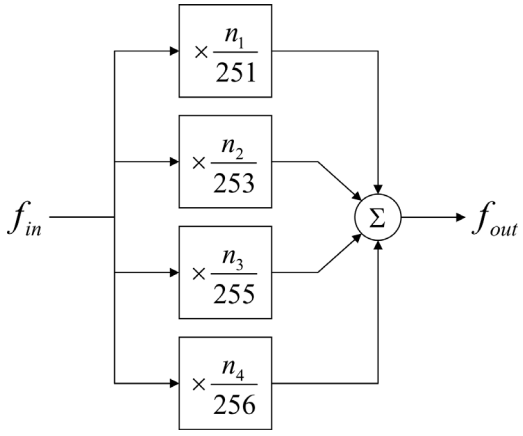

Fig. 6. Four PLLs DFS scheme. Constants $\bar{n}_1$ to $\bar{n}_4$ are omitted for simplicity.

TABLE II
FREQUENCIES SYNTHESIZED BY THE DFS ARCHITECTURE OF FIG. 6.

$$\frac{n_1}{251} + \frac{n_2}{253} + \frac{n_3}{255} + \frac{n_4}{256} = \frac{a}{4,145,475,840}$$

n_1	n_2	n_3	n_4	a
−251	0	0	0	−4,145,475,840
−69	−232	32	17	−4,145,475,839
−138	−211	64	34	−4,145,475,838
⋮	⋮	⋮	⋮	⋮
−226	−84	127	188	−4
−44	−63	−96	205	−3
−113	−42	−64	222	−2
−182	−21	−32	239	−1
0	0	0	0	0
−69	21	32	17	1
−138	42	64	34	2
−207	63	96	51	3
−25	−169	128	68	4
⋮	⋮	⋮	⋮	⋮
−113	−42	191	222	4,145,475,838
−182	−21	223	239	4,145,475,839
251	0	0	0	4,145,475,840

Consider the synthesizer in Fig. 6 using four PLLs, each having a prescaler less than or equal to 256. Therefore, the frequency resolution of any of the individual PLLs is not better than $f_{\text{in}}/256 \approxeq 4 \cdot 10^{-3} f_{\text{in}}$.

In contrast, as shown in Table II, the frequency resolution (step) of the whole architecture is

$$\frac{f_{\text{in}}}{251 \cdot 253 \cdot 255 \cdot 256} = \frac{f_{\text{in}}}{4,145,475,840} \approxeq 2.4 \cdot 10^{-10} \cdot f_{\text{in}}.$$

More specifically, combinations of values of n_1, n_2, n_3, and n_4 within their ranges $-251 \le n_1 \le 251$, $-253 \le n_2 \le 253$, $-255 \le n_3 \le 255$, and $-256 \le n_4 \le 256$ can generate all frequencies of the form

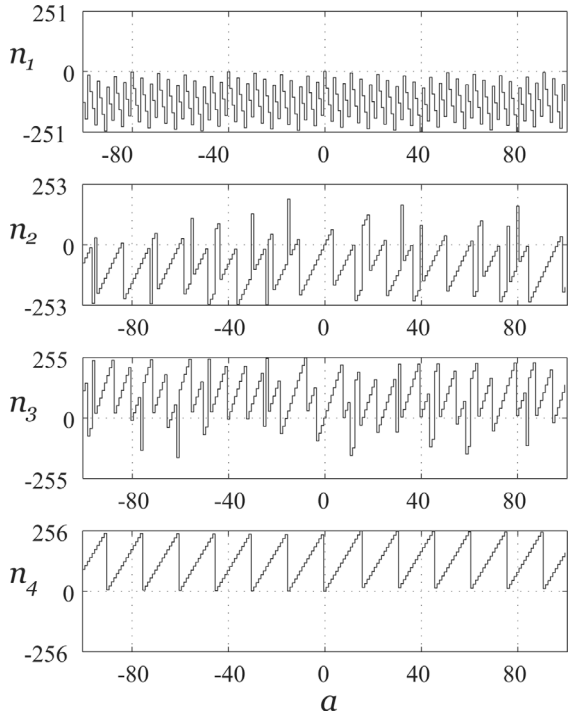

Fig. 7. Solution (n_1, n_2, n_3, n_4) of the Diophantine equation $n_1/251 + n_2/253 + n_3/255 + n_4/256 = a/(251 \cdot 253 \cdot 255 \cdot 256)$ as parameter a varies from -100 to 100.

$$f_{\text{out}} = \frac{a}{4,145,475,840}\, f_{\text{in}}, \tag{7}$$

where $-4,145,475,840 \le a \le 4,145,475,840$.

To emphasize this further, let's assume $f_{\text{in}} = 1$ MHz. Then, the output frequency range would be: -1 MHz to $+1$ MHz around $\overline{f_{\text{out}}}$ (which is zero here for simplicity). The frequency resolution would be 240 μHz. Recall that all prescalers are smaller than or equal to 256 and the frequency resolutions of the individual PLLs are about 4 kHz, i.e, more than 16 million times larger than that of the synthesizer.

To get the output frequency (7) corresponding to a particular value of a, one has to solve the linear Diophantine equation (8) and derive the values of n_1, n_2, n_3, and n_4.

$$\frac{n_1}{251} + \frac{n_2}{253} + \frac{n_3}{255} + \frac{n_4}{256} = \frac{a}{4,145,475,840} \tag{8}$$

As is proven in Section V, this can always be done within the assumed ranges of n_1, n_2, n_3, and n_4. The solution is not unique for certain values of a.

Fig. 7 shows a solution (quadruple) of (8) for a in the interval $-100, -99, -98, \ldots, 100$.

IV. The General Case

As briefly discussed in the previous sections, the ability to achieve very small frequency steps while using small size prescalers, $N_1, N_2, \ldots, N_k$, at the same time is based on the following fact:

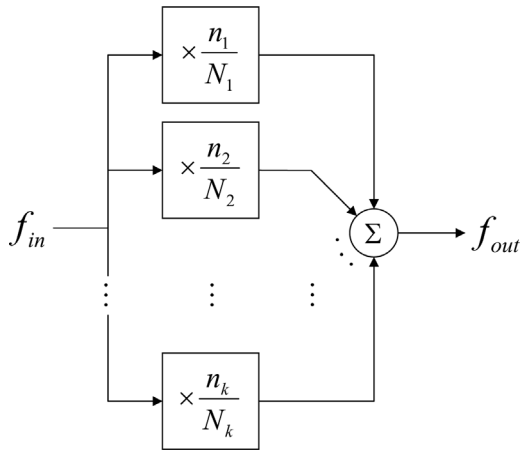

Fig. 8. k-PLLs DFS scheme. Parameters $\bar{n}_1$ to $\bar{n}_k$ are omitted.

If $N_1, N_2, \ldots, N_k$ are pairwise relatively prime then for every integer a, we can find integers $n_1, n_2, \ldots, n_k$ such that $\frac{n_1}{N_1} + \frac{n_2}{N_2} + \cdots + \frac{n_k}{N_k} = \frac{a}{N_1 N_2 \cdots N_k}$.

Therefore, the architecture in Fig. 8 provides frequency step equal to $f_{\text{in}}/(N_1\, N_2\, N_3 \cdots N_k)$. In some sense we can say that by using two or more PLLs we "distribute" the frequency resolution among them.

Note that the aforementioned frequency steps would not be achievable if the prescalers $N_1, N_2, \ldots, N_k$ were not pairwise relatively prime.

The following section provides the formal proof of the above discussion along with numerical algorithms for solving the Diophantine equation.

V. The Mathematical Framework of Diophantine Frequency Synthesis

This section provides the mathematical support for the general DFS scheme of Fig. 8.

Proposition 5.1: If $N_1, N_2, \ldots, N_k$ are pairwise relatively prime positive integers, then

$$\gcd\left(\prod_{\substack{i=1\\ i\neq 1}}^{k} N_i,\ \prod_{\substack{i=1\\ i\neq 2}}^{k} N_i,\ \ldots,\ \prod_{\substack{i=1\\ i\neq k}}^{k} N_i\right) = 1. \tag{9}$$

Proof: Suppose, in the contrary, that there exists a positive integer, not equal to one, that divides all products in (9). Then there must exist a prime number p with the same property. For $j = 1, 2, \ldots, k$, p divides the product $\prod_{i\neq j} N_i$, and so from Euclid's lemma [9], p must also divide at least one of the multiplicands, let N_{i_j} be one of them. If $N_{i_1}, N_{i_2}, \ldots, N_{i_k}$ are not all equal, then $N_1, N_2, \ldots, N_k$ cannot be pairwise relatively prime, i.e., a contradiction. If they are all equal, i.e., $N_{i_1} = N_{i_2} = \cdots = N_{i_k}$, they must also equal N_m for some $m \in \{1, 2, \ldots, k\}$. However, this is not possible since N_m is not in the m^{th} product and N_m does not have a nontrivial common divider with any other multiplicand in the m^{th} product. □

Lemma 5.1: Let $N_1, N_2, \ldots, N_k$ be pairwise relatively prime positive integers. Then, for every integer a, there exists a k-tuple of integers $(x_1, x_2, \ldots, x_k)$, solving the linear Diophantine equation

$$\frac{x_1}{N_1} + \frac{x_2}{N_2} + \cdots + \frac{x_k}{N_k} = \frac{a}{N_1 N_2 \cdots N_k}. \tag{10}$$

Proof: For $j = 1, 2, \ldots, k$, we set $E_j = \prod_{i \neq j} N_i$. Multiplying (10) by $N_1 N_2 \cdots N_k$ gives the equivalent (11).

$$E_1 x_1 + E_2 x_2 + \cdots + E_k x_k = a. \tag{11}$$

From the assumptions of Lemma 5.1 and Proposition 5.1 we get that $\gcd(E_1, E_2, \ldots, E_k) = 1$. This is a sufficient and necessary condition for the Diophantine equation (11) to have a solution for every integer a [9]. □

So far we have proven that Diophantine equation (10) has a solution $(x_1, x_2, \ldots, x_k)$ but we know nothing about the ranges of x_i's. The following theorem provides an answer.

Theorem 5.1: If $N_1, N_2, \ldots, N_k$ are pairwise relatively prime positive integers, then for every integer a such that $-N_1 N_2 \cdots N_k \leq a \leq N_1 N_2 \cdots N_k$, the Diophantine equation (10) has a solution $(x_1, x_2, \ldots, x_k)$, where $-N_i \leq x_i \leq N_i$ for all $i = 1, 2, \ldots, k$.

Proof: If $a = \pm N_1 N_2 \cdots N_k$, such a solution of (10) is given by $x_1 = \pm N_1$ and $x_i = 0$ for $i = 2, 3, \ldots, k$, respectively. Now suppose that a is absolutely smaller than $N_1 N_2 \cdots N_k$ and let $(z_1, z_2, \ldots, z_k)$ be a (any) solution of (10), i.e.,

$$\frac{z_1}{N_1} + \frac{z_2}{N_2} + \cdots + \frac{z_k}{N_k} = \frac{a}{N_1 N_2 \cdots N_k}. \tag{12}$$

Set $y_i = z_i \mod N_i$ for $i = 1, 2, \ldots, k$. Then, by the definition of the y_i's, there exists an integer q such that

$$\frac{y_1}{N_1} + \frac{y_2}{N_2} + \cdots + \frac{y_k}{N_k} = \frac{a}{N_1 N_2 \cdots N_k} + q. \tag{13}$$

Since $0 \leq y_i < N_i$ for $i = 1, 2, \ldots, k$, it is

$$0 \leq \frac{y_1}{N_1} + \frac{y_2}{N_2} + \cdots + \frac{y_k}{N_k} < k. \tag{14}$$

By combining (14) with (13) we get

$$-\frac{a}{N_1 N_2 \cdots N_k} \leq q < k - \frac{a}{N_1 N_2 \cdots N_k}. \tag{15}$$

Since we have assumed that $|a| < N_1 N_2 \cdots N_k$, from (15) we conclude that $-1 < q < k + 1$, which implies

$$q \in \{0, 1, \ldots, k\}.$$

If $q = 0$, then set $x_i = y_i$, $i = 1, 2, \ldots, k$. If $q > 0$, then set $x_i = y_i - N_i$ for $i = 1, 2, \ldots, q$ and $x_i = y_i$ for $i = q + 1, \ldots, k$. In both cases it is $-N_i \leq x_i \leq N_i$ for all $i = 1, 2, \ldots, k$. □

Fig. 9. Although a can achieve values beyond $\pm N_1 N_2 \ldots N_k$, when $|n_i| \leq N_i$, $i = 1, 2, \ldots, k$, the step $1/(N_1 N_2 \cdots N_k)$ is not guaranteed.

Note that Theorem 5.1 guarantees only the existence of at least one solution within the specified bounds, $|x_i| \leq N_i$ for all indices i, but not the uniqueness of it. In Table I, for example, we see that equation $n_1/3 + n_2/5 = -1/15$ has (at least) two solutions: $-2/3 + 3/5 = 1/3 - 2/5 = -1/15$.

Also, Theorem 5.1 guarantees the existence of solution within the specified bounds when $|a| \leq N_1 N_2 \ldots N_k$, but it does *not* say that a solution within those bounds cannot be found for values of a that are absolutely larger than $N_1 N_2 \ldots N_k$. Following the previous examples, we see that $3/3 + 2/5 = 21/15 > 1$ and $200/251 + 180/253 + 210/255 + 190/256 = 3 + 306732510/4145475840$.

The problem in general is that for $|a| > N_1 N_2 \ldots N_k$, the step size $1/(N_1 N_2 \cdots N_k)$ is not guaranteed when the constraints $|n_i| \leq N_i$, $i = 1, 2, \ldots, k$ are satisfied.

Consider, for example, the Diophantine equation $n_1/3 + n_2/16 = a/48$ but now think of a as a function of n_1 and n_2. All values that a takes, when n_1 and n_2 range within $|n_1| \leq 3$ and $|n_2| \leq 16$, respectively, are shown in Fig. 9. The white (vertical) gaps correspond to values of a that cannot be synthesized.

Especially for larger values of $N_1, N_2, \ldots, N_k$, expansion of the range of a beyond $\pm N_1 N_2 \ldots N_k$ is practically insignificant.

Interpretation of Theorem 5.1: Rephrasing Theorem 5.1, we can say that given a set $N_1, N_2, \ldots, N_k$ of pairwise relatively prime positive integers, **all** rational numbers from -1 to 1 with uniform step (resolution) $1/(N_1 N_2 \cdots N_k)$ are generated by the sum $x_1/N_1 + x_2/N_2 + \cdots + x_k/N_k$ when the numerators $x_1, x_2, \ldots, x_k$ vary within the intervals $-N_i \leq x_i \leq N_i$, $i = 1, 2, \ldots, k$. The practical value of this statement is summarized in the sentence:

A very high resolution of a parameter can be achieved by controlling the values of a set of parameters with very low resolution.

Example 5.1: Consider the pairwise relatively prime numbers 11, 16, 17, 19, and 23. According to Theorem 5.1, the sum of the fractions

$$\frac{x_1}{11} + \frac{x_2}{16} + \frac{x_3}{17} + \frac{x_4}{19} + \frac{x_5}{23} \tag{16}$$

takes (at least) all values from -1 to $+1$ with resolution equal to $1/(11 \cdot 16 \cdot 17 \cdot 19 \cdot 23) < 10^{-6}$ when the numerators vary within the ranges $-11 \leq x_1 \leq 11$, $-16 \leq x_2 \leq 16$, $-17 \leq x_3 \leq 17$, $-19 \leq x_4 \leq 19$, and $-23 \leq x_5 \leq 23$. □

Example 5.2: Expanding the sum in expression (16) by four additional fractions with denominators 29, 31, 37, and 41 (note that the set of all denominators is formed by pairwise relatively prime integers) we get

$$\frac{x_1}{11} + \frac{x_2}{16} + \frac{x_3}{17} + \frac{x_4}{19} + \frac{x_5}{23} + \frac{x_6}{29} + \frac{x_7}{31} + \frac{x_8}{37} + \frac{x_9}{41}. \tag{17}$$

Eq. (17) takes (at least) all values from -1 to $+1$ with resolution better (smaller) than 10^{-12}. □

A. Particular Solutions of the Diophantine Equations

The design of a Diophantine frequency synthesizer involves the solution of the linear Diophantine equation (10). If only *one* output frequency is desirable, then (10) must be solved once for a particular value of a only. If variable output frequency is desirable, then (10) must be solved for all values of a that may be used. Solving the equation requires some computational effort and, in most cases, large integer number algebraic manipulation.

To avoid this computational complexity, one could consider storing the solutions corresponding to all possible values a to a memory device, and use them to program the dividers of the PLLs when needed. Although this could be done for certain cases, in general it may require a large amount of storage space; consider, for example, the DFS scheme in Fig. 6 where about 8 billion quadruples must be stored.

Alternatively, one can use the solution of the particular eq. (18), i.e., when $a = 1$, to generate, in a very simple way, the solution of (10) for every value of a. The procedure is given by the following lemma.

Lemma 5.2: Let $(z_1, z_2, \ldots, z_k)$ be a solution of the Diophantine equation

$$\frac{z_1}{N_1} + \frac{z_2}{N_2} + \cdots + \frac{z_k}{N_k} = \frac{1}{N_1 N_2 \cdots N_k}. \tag{18}$$

Then, a solution $(x_1, x_2, \ldots, x_k)$ of the general Diophantine equation (10) with $-N_i \le x_i \le N_i$, $i = 1, 2 \ldots, k$, can be found in the following way:

- If $a = N_1 N_2 \cdots N_k$, then set $x_1 = N_1$ and $x_i = 0$ for $i = 2, 3, \ldots, k$.
- If $a = -N_1 N_2 \cdots N_k$, then set $x_1 = -N_1$ and $x_i = 0$ for $i = 2, 3, \ldots, k$.
- If $-N_1 N_2 \cdots N_k < a < N_1 N_2 \cdots N_k$, then set $y_i = a z_i \mod N_i$, $i = 1, 2 \ldots, k$ and calculate

$$q = \frac{y_1}{N_1} + \frac{y_2}{N_2} + \cdots + \frac{y_k}{N_k} - \frac{a}{N_1 N_2 \cdots N_k}$$

  Finally, set $x_i = y_i - N_i$ for $i = 1, 2, \ldots, q$, and $x_i = y_i$ for $i = q + 1, \ldots, k$.

The proof of Lemma 5.2 is very similar to that of Theorem 5.1 and is omitted. Moreover, from the same proof, we have that q is a nonnegative integer, less than or equal to k, and so the steps above are valid.

Example 5.3: The methodology of Lemma 5.2 is used to solve Diophantine equation (19) for a ranging within the interval $-6 \le a \le 6$, using the particular solution $1/2 + (-1)/3 = 1/6$. i.e., $(z_1, z_2) = (1, -1)$.

$$\frac{x_1}{2} + \frac{x_2}{3} = \frac{a}{6}. \tag{19}$$

TABLE III
APPLICATION OF LEMMA 5.2 FOR $k = 2$, $N_1 = 2$, AND $N_2 = 3$, USING THE PARTICULAR SOLUTION $(z_1, z_2) = (1, -1)$.

a	$y_1 = az_1 \mod 2$	$y_2 = az_2 \mod 3$	q	x_1	x_2
−6	—	—	—	−2	0
−5	1	2	2	−1	−1
−4	0	1	1	−2	1
−3	1	0	1	−1	0
−2	0	2	1	−2	2
−1	1	1	1	−1	1
0	0	0	0	0	0
1	1	2	1	−1	2
2	0	1	0	0	1
3	1	0	0	1	0
4	0	2	0	0	2
5	1	1	0	1	1
6	—	—	—	2	0

Table III shows the values of a, x_1, and x_2, as well as those of the intermediate variables y_1, y_2, and q (see Lemma 5.2). □

Tables I and II were generated using the procedure of Lemma 5.2 and the particular solutions $(-1)/3 + 2/5 = 1/15$ and $(-69)/251 + 21/253 + 32/255 + 17/256 = 1/4,145,475,840$, respectively.

B. Derivation of a Particular Solution

The foundation of DFS lies in the (existence and derivation of the) solutions of (10) and (18). In this section it is shown that their solutions can be derived by solving $k - 1$ linear Diophantine equations of two variables (only). The last one can be done easily, e.g., using MATLAB.

The following well-known theorem is stated without proof; it can be found in textbooks on number theory, e.g., [9].

Theorem 5.2: For nonzero integers m_1, m_2 and d, the Diophantine equation $m_1 x_1 + m_2 x_2 = d$ has a solution if and only if $\gcd(m_1, m_2)$ divides d. If a solution exists, it can be derived using the Euclidean algorithm [9].

For most practical purposes, if $\gcd(m_1, m_2)$ divides d, equation $m_1 x_1 + m_2 x_2 = d$ can be solved using MATLAB's command "gcd." Specifically, executing "$[g, y_1, y_2] = \gcd(m_1, m_2)$" in MATLAB returns g, y_1, and y_2 such that $g = \gcd(m_1, m_2)$ and $m_1 y_1 + m_2 y_2 = g$ and so, $x_1 = (d/g)\, y_1$ and $x_2 = (d/g)\, y_2$. Therefore, the case of (18) with $k = 2$ can be addressed easily.

Now consider (18) with $k = 3$. By multiplying both sides with $N_1 N_2 N_3$ we can write

$$N_3 (N_2 z_1 + N_1 z_2) + N_1 N_2 x_3 = 1. \tag{20}$$

Also note that N_1, N_2, N_3 are pairwise relatively prime, so $\gcd(N_2, N_1) = 1$ and $\gcd(N_3, N_1 N_2) = 1$. Therefore, according to Theorem 5.2, we can find a solution, (w_1, w_2),

of $N_2w_1 + N_1w_2 = 1$, and a solution, (w, z_3), of $N_3w + N_1N_2z_3 = 1$. Then we have

$$\begin{aligned} 1 &= N_3w + N_1N_2z_3 \\ &= N_3(N_2w_1 + N_1w_2)w + N_1N_2z_3 \\ &= N_3N_2(ww_1) + N_3N_1(ww_2) + N_1N_2z_3, \end{aligned}$$

and so $(z_1, z_2, z_3) = (ww_1, ww_2, z_3)$ is a solution of (20).

The procedure extends naturally for $k > 3$, and it is stated in the following lemma without proof.

Lemma 5.3: A solution of (18) is derived by solving the following $k-1$ linear Diophantine equations of two variables:

$$\begin{aligned} N_2z_2 + N_1w_2 &= 1 \\ N_3z_3 + N_1N_2w_3 &= 1 \\ N_4z_4 + N_1N_2N_3w_4 &= 1 \\ \vdots \qquad\qquad &\vdots \\ N_kz_k + N_1N_2N_3\cdots N_{k-1}w_k &= 1, \end{aligned} \tag{21}$$

and then setting

$$\begin{aligned} x_1 &= z_2z_3z_4\cdots z_k \\ x_r &= z_{r+1}z_{r+2}\cdots z_kw_r \\ &\qquad r = 2, 3, \ldots, k-1 \\ x_k &= w_k. \end{aligned} \tag{22}$$

Finally, a solution of (10) is derived using the algorithm in Lemma 5.2. The algorithm, implemented in MATLAB, is given in Appendix B.

VI. Fixed Frequency DFS: An Example

In many situations it is desirable to generate a periodic signal of a specific and fixed frequency f_{out} using a reference signal at a given frequency f_{in}. This is typical in atomic clocks and related time reference systems. Let's consider the following example:

The input frequency is $f_{in} = 10$ MHz and the desirable output frequency is $f_{out} = 9.285,739,4$ MHz which must be synthesized with accuracy of 0.1 Hz.

To achieve 0.1 Hz resolution with only one PLL, a prescaler equal to or greater than $f_{\text{in}}/0.1\,\text{Hz} = 10^8$ is required. This is definitely impractical for most realistic situations. Although other techniques can be used to achieve this resolution [1], [2], the Diophantine approach is straightforward. Two scenarios are presented using combinations of two and three basic PLLs, respectively.

A. Two PLLs DFS Scheme

Let the prescalers of the two PLLs be N_1 and N_2. Moreover, let's assume for simplicity that $N_1 \approxeq N_2$.

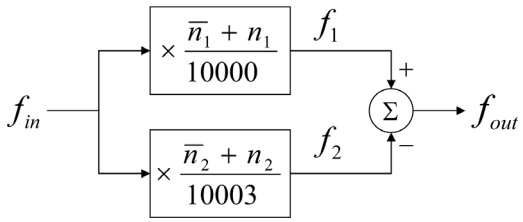

Fig. 10. Two PLLs DFS scheme.

From Section V we know that the output frequency resolution of the synthesizer is $f_{\text{in}}/(N_1N_2)$. Since accuracy of 0.1 Hz, or better, is required while the input frequency is 10 MHz, it must be that $N_1N_2 \geq f_{\text{in}}/0.1\text{ Hz} = 10^8$. Therefore we can choose $N_1, N_2 \approxeq \sqrt{10^8} = 10^4$, assuming that we want to keep the prescalers as small as possible.

We can pick, for example, the pair of relatively prime integers $N_1 = 10,000$ and $N_2 = 10,003$ with $N_1N_2 = 100,030,000$. Then the phase-comparator frequencies of the PLLs are about 1 kHz. A DFS scheme based on these prescalers is shown in Fig. 10.

For now, we ignore $\bar{n}_1$ and $\bar{n}_2$ ($\bar{n}_1 = \bar{n}_2 = 0$) and focus our attention on tuning f_{out} using n_1 and n_2. From Section V we know that by choosing appropriate values for n_1 and n_2 (and ignoring $\bar{n}_1$ and $\bar{n}_2$), f_{out} can take any value

$$f_{\text{out}} = \frac{n}{100,030,000}\, f_{\text{in}}, \tag{23}$$

where n ranges from −100,030,000 to 100,030,000.

Choosing $n = 92,885,251$ we get output frequency $f_{\text{out}} = 9.285,739,378\ldots$. This is the best possible approximation by (23) to the desirable frequency 9.285,739,4 MHz, and the frequency error is smaller than 0.1 Hz.

Now we derive the values of n_1 and n_2 resulting in $n = 92,885,251$. To do so we must solve the Diophantine equation

$$\frac{n_1}{10000} + \frac{-n_2}{10003} = \frac{n}{100,030,000}. \tag{24}$$

Note that the minus sign (due to frequency mixing in the scheme of Fig. 10) does not cause any complication since the ranges of n_1 and n_2 are symmetric with respect to zero. We simply solve (24) for n_1 and $(-n_2)$.

To proceed, we use the "gcd" function of MATLAB[5]. It gives "gcd(10003, 10000) = [1, −3333, 3334]" and so

$$\frac{-3333}{10000} + \frac{3334}{10003} = \frac{1}{100,030,000}.$$

Following Lemma 5.2, we set (where $n = 92,885,251$)

$$\begin{aligned} y_1 &= (-3333 \cdot n) \mod 10000 \\ &= 8417, \\ y_2 &= (3334 \cdot n) \mod 10003 \\ &= 869. \end{aligned}$$

[5]Note the reversed order of the arguments in gcd.

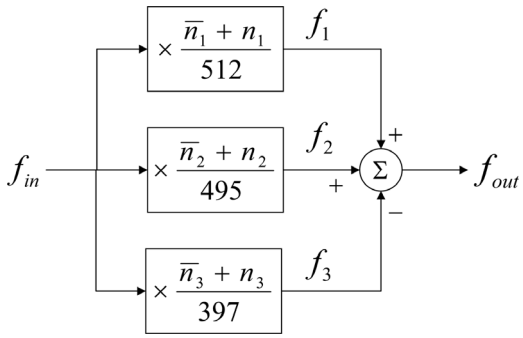

Fig. 11. Three PLLs DFS scheme.

Since

$$\frac{y_1}{10000} + \frac{y_2}{10003} = \frac{92,885,251}{100,030,000},$$

is smaller than 1, it is $q = 0$ in Lemma 5.2, and so the pair (y_1, y_2) is the desirable solution. Therefore, $n_1 = 8417$, $(-n_2) = 869$, and

$$\left(\frac{8417}{10000} + \frac{-869}{10003}\right) \cdot 10 \text{ MHz} = 9.285,739,378\ldots \text{MHz}.$$

Now we concentrate on $\bar{n}_1$ and $\bar{n}_2$. Note that

$$f_{\text{out}} = \left(\frac{\bar{n}_1}{N_1} - \frac{\bar{n}_2}{N_2}\right) f_{\text{in}} + \left(\frac{n_1}{N_1} - \frac{n_2}{N_2}\right) f_{\text{in}},$$

and since the second summand equals the desirable frequency, the first summand must be zero. For this to happen, it must be $\bar{n}_i = c\,N_i$, $i = 1, 2$, where c is an integer. The proof is given for the general case in Lemma 9.1 in Appendix A. Moreover, c must be positive because the frequency multiplication ratio of the i^{th} PLL, which equals $(\bar{n}_i + n_i)/N_i = c + n_i/N_i$, is always positive.

The value of c can be chosen to minimize the mixing spurs [1], minimize the phase noise introduced by the VCOs, or optimize the circuit otherwise. One choice could be $c = 5$, which implies

$$f_1 = \frac{\bar{n}_1 + n_1}{N_1} f_{\text{in}} \cong 58.417,000,0 \text{ MHz}$$
$$f_2 = \frac{\bar{n}_2 + n_2}{N_2} f_{\text{in}} \cong 49.131,260,6 \text{ MHz}$$

and therefore a ratio f_2/f_1 close to 0.85, resulting in low mixing spurs [1].

B. Three PLLs DFS Scheme

Use of three PLLs allows for more flexibility. Let's assume again that $N_1 \cong N_2 \cong N_3$ which implies the minimal values $N_i \cong \sqrt[3]{10^8} \cong 464$. An appropriate triplet of pairwise relatively prime integers is $N_1 = 512$, $N_2 = 495$, and $N_3 = 397$, giving $N_1 N_2 N_3 = 100,615,680$.

A DFS scheme using these numbers is shown in Fig. 11. Let's ignore the constants $\bar{n}_1$, $\bar{n}_2$, and $\bar{n}_3$ for the moment (consider $\bar{n}_1 = \bar{n}_2 = \bar{n}_3 = 0$ for now). Then, by adjusting n_1, n_2, and n_3, the output frequency can take any of the values

$$f_{\text{out}} = \frac{n}{100,615,680} f_{\text{in}}, \tag{25}$$

where n ranges from $-100,615,680$ to $100,615,680$. The best approximation of the desirable frequency, 9,285,739,4 MHz, with $f_{\text{in}} = 10$ MHz, is achieved using $n = 93,429,098$. Now we have to solve (26) for n_1, n_2, and n_3. The minus sign in n_3 is due to frequency mixing in the scheme of Fig. 11.

$$\frac{n_1}{512} + \frac{n_2}{495} + \frac{-n_3}{397} = \frac{93,429,098}{100,615,680}. \tag{26}$$

Using the algorithm in Lemma 5.3, we get

$$\frac{-501}{512} + \frac{134}{495} + \frac{281}{397} = \frac{1}{100,615,680},$$

and following the procedure of Lemma 5.2, we get

$$\frac{-114}{512} + \frac{217}{495} + \frac{283}{397} = \frac{93,429,098}{100,615,680}. \tag{27}$$

So $n_1 = -114$, $n_2 = 217$, and $n_3 = -283$. Now taking $\bar{n}_1$, $\bar{n}_2$, and $\bar{n}_3$ into account we have

$$f_{\text{out}} = \left(\frac{\bar{n}_1}{N_1} + \frac{\bar{n}_2}{N_2} - \frac{\bar{n}_3}{N_3}\right) f_{\text{in}} + \frac{93,429,098}{100,615,680} f_{\text{in}}.$$

As in the previous example, we can choose the values of $\bar{n}_1$, $\bar{n}_2$, and $\bar{n}_3$ to minimize mixing spurs or noise, or bring f_1, f_2, and f_3 within the operating range of existing PLLs, or optimize some other criterion. However, (here) we would like to do so without changing f_{out} since it already has the desirable value, therefore it must be

$$\frac{\bar{n}_1}{N_1} + \frac{\bar{n}_2}{N_2} - \frac{\bar{n}_3}{N_3} = 0. \tag{28}$$

Since N_1, N_2, and N_3 are pairwise relatively prime, (28), along with Lemma 9.1 in Appendix A, imply that $\bar{n}_1 = c_1 N_1$, $\bar{n}_2 = c_2 N_2$, and $\bar{n}_3 = c_3 N_3$ with $c_1 + c_2 - c_3 = 0$.

An eligible choice, for example, is $c_1 = 1$, $c_2 = 1$, and $c_3 = 2$. This gives $f_1 = 7.773,437,50$ MHz, $f_2 = 14.383,838,38$ MHz, and $f_3 = 12.871,536,52$ MHz.

VII. Variable Frequency DFS: An Example

Suppose we want to design a DFS synthesizer that can generate frequencies from 2 MHz to 4 MHz with resolution of about 1 Hz. From the theory Section V we know that the general architecture of Fig. 8, with input frequency f_{in}, can generate all frequencies from $\overline{f_{\text{out}}} - f_{\text{in}}$ to $\overline{f_{\text{out}}} + f_{\text{in}}$ with resolution $f_{\text{in}}/(N_1 N_2 \cdots N_k)$. Since the frequency range is $2f_{\text{in}}$, we can choose

$$f_{\text{in}} = 1 \text{ MHz}. \tag{29}$$

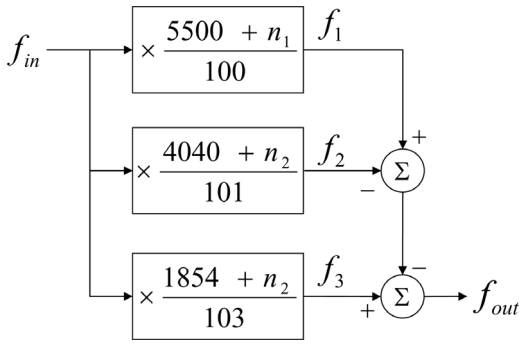

Fig. 12. 3-PLLs variable frequency DFS scheme.

Then, the resolution requirement is satisfied if

$$N_1 N_2 \cdots N_k \geq \frac{f_{\text{in}}}{1 \text{ Hz}} = 10^6. \tag{30}$$

Suppose we add the requirement that the phase-comparator frequencies in all PLLs are about 10 kHz. This means that

$$\frac{f_{\text{in}}}{N_i} \cong 10 \text{ kHz}, \quad i = 1, 2, \ldots, k, \tag{31}$$

which implies $N_i \cong 100$ and $N_1 N_2 \ldots N_k \cong 100^k$. Therefore, from (30), the minimum number of PLLs, k, we should use is $k = 3$. Three pairwise relatively prime numbers are $N_1 = 100$, $N_2 = 101$, and $N_3 = 103$.

The next step is to decide what the central frequencies of the three PLLs should be and how they will be mixed, i.e., added or subtracted. Since the purpose of this paper is solely to present the mathematical principles of DFS, many technical issues (e.g., the pullability range of the PLLs, the spurs generated by the mixing, the possible filtering of the PLLs' signals before mixing, the minimization of the output phase noise, etc.) involved in these decisions are not discussed here.

A simple choice[6] is $\overline{f_1} = 55$ MHz, $\overline{f_2} = 40$ MHz, and $\overline{f_3} = 18$ MHz and the output frequency is chosen to be $f_{\text{out}} = -(f_1 - f_2) + f_3$, resulting in $\overline{f_{\text{out}}} = 3$ MHz. Since $\overline{f_i} = (\bar{n}_i / N_i) f_{\text{in}}$, $i = 1, 2, 3$, we have $\bar{n}_1 = 5500$, $\bar{n}_2 = 4040$, and $\bar{n}_3 = 1854$. The corresponding DFS architecture is shown in Fig. 12.

The frequency ranges of the PLLs and of the output signal, along with their resolutions, are shown in Table IV. The output frequency can take all values

$$f_{\text{out}} = \left(3 + \frac{n}{1,040,300}\right) \text{ MHz},$$

where n ranges from −1,040,300 to 1,040,300.

Given the desirable value of n, parameters n_1, n_2, and n_3 are derived using Lemmas 5.2 and 5.3. Specifically, the algorithm in Lemma 5.3 gives

$$\frac{-33}{100} + \frac{-51}{101} + \frac{86}{103} = \frac{1}{1,040,300}. \tag{32}$$

[6]No effort has been made to optimize this choice. Using more elaborate mixing schemes, one can possibly reduce $\overline{f_1}, \overline{f_2}$, and $\overline{f_3}$ while maintaining a clean output spectrum.

TABLE IV
FREQUENCY RANGES AND FREQUENCY STEPS (RESOLUTIONS) OF THE SIGNALS IN THE DFS SCHEME OF FIG. 12.*

	Min	Central	Max	Frequency step (resolution)
f_{in}	—	1	—	—
f_1	54	55	56	1/100
f_2	39	40	41	1/101
f_3	17	18	19	1/103
f_{out}	2	3	4	1/1,040,300

*All frequencies are in MHz.

Following Lemma 5.2, we set

$$\begin{aligned} y_1 &= (-33 \cdot n) \mod 100 \\ y_2 &= (-51 \cdot n) \mod 101 \\ y_3 &= (86 \cdot n) \mod 103 \end{aligned}$$

and calculate the value of q using

$$q = \frac{y_1}{100} + \frac{y_2}{101} + \frac{y_3}{103} - \frac{n}{1,040,300}.$$

Recall from Lemma 5.2 that since $k = 3$, q can take only one of the values 0, 1, 2, or 3. Depending on q, we set:

$$\begin{array}{lll} & n_1 = -y_1 & \\ \text{Either} & n_2 = \ \ y_2 & \text{if } q = 0 \\ & n_3 = \ \ y_3 & \\ & & \\ & n_1 = -(y_1 - N_1) & \\ \text{or} & n_2 = \ \ y_2 & \text{if } q = 1 \\ & n_3 = \ \ y_3 & \\ & & \\ & n_1 = -(y_1 - N_1) & \\ \text{or} & n_2 = \ \ y_2 - N_2 & \text{if } q = 2 \\ & n_3 = \ \ y_3 & \\ & & \\ & n_1 = -(y_1 - N_1) & \\ \text{or} & n_2 = \ \ y_2 - N_2 & \text{if } q = 3 \\ & n_3 = \ \ y_3 - N_3 & \end{array}$$

The minus sign of n_1 is due to the frequency mixing in the scheme of Fig. 12, i.e., $f_{\text{out}} = -f_1 + f_2 + f_3$.

VIII. CONCLUSIONS

The Diophantine Frequency Synthesis (DFS) approach for fine frequency synthesis was introduced. It is based on number theory, it uses two or more basic PLLs, and allows for independent choices of the output frequency step (resolution) and the phase-comparator frequencies in the PLLs.

DFS leads to fine frequency step, fast frequency hopping architectures with potentially very low spurs, especially in the vicinity of the carrier.

The paper focused on the mathematical principles of DFS and the related algorithms.

Appendix A

Lemma 9.1: If $N_1, N_2, \ldots, N_k$ are pairwise relatively prime integers, then every solution of (33) is of the form $x_i = c_i N_i$, $i = 1, 2, \ldots, k$, where $c_1, c_2, \ldots, c_k$ are integers such that $c_1 + c_2 + \cdots + c_k = 0$.

$$\frac{x_1}{N_1} + \frac{x_2}{N_2} + \cdots + \frac{x_k}{N_k} = 0 \tag{33}$$

Proof: Multiplying (33) by $N_1 N_2 \cdots N_k$ gives

$$E_1 x_1 + E_2 x_2 + \cdots + E_k x_k = 0, \tag{34}$$

where we have set $E_j = \prod_{i \neq j} N_i$ for $j = 1, 2, \ldots, k$. Since N_i divides E_j for all $j \neq i$, from (34) there exists an integer m_i such that

$$m_i N_i + E_i x_i = 0. \tag{35}$$

We also have that $\gcd(N_i, E_i) = 1$ since $N_1, N_2, \ldots, N_k$ are pairwise relatively prime. Therefore, N_i must divide x_i. This is true for all i's, and so there exist integers $c_1, c_2, \ldots, c_k$ such that $x_i = c_i N_i$ for $i = 1, 2, \ldots, k$. Replacing them in (34) we get $c_1 + c_2 + \cdots + c_k = 0$. □

Appendix B

The MATLAB algorithm below solves the particular Diophantine equation (18). When applying it, one should pay attention to the size of the integers involved. For large integers $N_1, N_2, \ldots, N_k$ and/or for large k, the calculations should be done using Variable Precision Arithmetic (VPA) or the algorithm should be restructured appropriately.

```
% ———————— Initialize ————————
N = [N_1, N_2, ..., N_k];
k = length (N);
x = zeros (1, k);
z = zeros (1, k);
w = zeros (1, k);
% ———————————————————————
%   For i = 2, 3, ..., k solve
%   N_i z_i + N_1 N_2 ··· N_{i-1} w_i = 1
% ———————————————————————
for i = 2 : k
   [g, z(i), w(i)] = gcd (N(i), prod (N(1 : i − 1)));
end
% ———————————————————————
%   Derive x_1, x_2, ..., x_k
% ———————————————————————
x(1) = prod (z(2 : k));
for r = 2 : k − 1
   x(r) = prod (z(r + 1 : k)) ∗ w(r);
end
x(k) = w(k);
% ———————————————————————
```

Acknowledgments

The author would like to thank Dr. Tom Krimigis, Dr. Marion L. Edwards, Gregory Weaver, Sheng Cheng, Wesley Millard, and Christopher Haskins as well as Dr. Paul Ostdiek and Bob Bokulic from the Johns Hopkins University Applied Physics Laboratory for their support and many technical discussions during this project.

Appreciation is also extended to Michael M. Driscoll from the Northrop Grumman Corporation for his encouragement and feedback on this work.

References

[1] V. Manassewitsch, *Frequency Synthesizers*. 3rd ed. New York: Wiley, 1987.
[2] W. F. Egan, *Frequency Synthesis by Phase Lock*. 2nd ed. New York: Wiley, 1999.
[3] B.-G. Goldberg, *Digital Techniques in Frequency Synthesis*. New York: McGraw-Hill, 1995.
[4] V. F. Kroupa, "Close to the carrier noise in DDS," presented at IEEE Int. Freq. Symp., 1996.
[5] S. Cheng, "Analysis and simulation of the DDS (direct digital synthesis) architecture," Applied Physics Laboratory, The Johns Hopkins University, Tech. Rep. SER-04-029, 2004.
[6] H. T. Nicholas and H. Samueli, "An analysis of the output spectrum of direct digital frequency synthesizers in the presence of phase accumulator truncation," in *Proc. 41st Annu. Freq. Contr. Symp.*, 1987, pp. 495–502.
[7] A. Torosyan and A. N. Willson, Jr., "Exact analysis of DDS spurs and SNR due to phase truncation and arbitrary phase-to-amplitude errors," in *Proc. IEEE Int. Freq. Contr. Symp.*, 2005, pp. 50–58.
[8] S. Cheng, J. R. Jensen, R. E. Wallis, and G. L. Weaver, "Further enhancements to the analysis of spectral purity in the application of practical direct digital synthesis," in *Proc. Int. Freq. Contr. Symp. Expo.*, 2004, pp. 462–470.
[9] D. E. Flath, *Introduction to Number Theory*. New York: Wiley, 1989.
[10] R. E. Best, *Phase-Locked Loops: Design, Simulation, and Applications*. 5th ed. New York: McGraw-Hill, 2003.

Paul P. Sotiriadis received the diploma in electrical engineering and computer science from the National Technical University of Athens (NTUA), Greece; the M.S. degree in electrical engineering from Stanford University, Stanford, CA; and the Ph.D. degree in electrical engineering and computer science from the Massachusetts Institute of Technology, Cambridge, MA, in May 2002.

Since June 2002 he has been an assistant professor with the Department of Electrical and Computer Engineering at The Johns Hopkins University, Baltimore, MD.

His research interests include design, optimization, and mathematical modeling of analog and mixed-signal circuits, RF and microwave circuits, frequency synthesis, and interconnect networks in deep-sub-micron technologies. He serves as an associate editor of the *IEEE Transactions on Circuits and Systems-II*.