Cascaded Diophantine Frequency Synthesis

Paul Peter Sotiriadis, *Member, IEEE*

***Abstract*—Cascaded Diophantine Frequency Synthesis[1] (CDFS) is an approach to high-resolution frequency synthesis based on the mathematical properties of integer numbers and Diophantine equations. CDFS can be implemented using two or more phase-locked loops (PLLs) and frequency mixing stages in a cascade topology. CDFS achieves frequency resolution arbitrarily finer than that of the constituent PLLs while maintaining their loop bandwidths and frequency hopping agility. CDFS results in intermediate signals with minimal frequency ranges in all frequency mixing stages, allowing for improved spectral purity and lower design complexity compared to the parallel form of its predecessor, Diophantine Frequency Synthesis (DFS). CDFS architectures are modularly structured and expandable. The paper introduces CDFS focusing on the mathematical and algorithmic aspects.**

***Index Terms*—Diophantine equation , Direct Digital Synthesis (DDS), frequency hopping, frequency synthesis, instrumentation, navigation systems, number theory, phase-locked loop (PLL), timing systems, wireless communications.**

I. Introduction

FREQUENCY synthesis is an essential element of modern electronic systems including wired and wireless communications, computers, radar, timing and navigation (GPS), scientific instrumentation and many other devices.

Several frequency synthesis architectures have been proposed and a rich collection can be found in [1]–[5].

In modern frequency synthesizers the output signal is typically phase-coherent to the input one and so a frequency synthesizer is essentially a frequency multiplier (with phase, frequency or amplitude modulation capabilities if required). It multiplies the input reference frequency f_{in} by a factor a. In practice, factor a is the ratio n/R of two positive integers, n and R.

There are many ways of implementing this frequency multiplication which give rise to several different classes of frequency synthesizers. E.g., Integer-N phase-locked loops (PLLs) realize it by direct division, by R, and phase-locked multiplication by n, [6]; Fractional-N PLLs generate a by rational interpolation between two positive rational numbers [3]; Direct Digital Synthesis (DDS) realizes the multiplication in the phase domain also by direct division by R and multiplication by n, [7].

These primal synthesizers are often used as building blocks to produce multiloop frequency synthesizers with superior performance that compose a as an algebraic sum of rational numbers, e.g., in the popular mix-and-divide approach, [1] it is

$$a = \frac{n_1}{R_1} \pm \frac{1}{Q_1}\left(\frac{n_2}{R_2} \pm \frac{1}{Q_2}(\cdots)\right).$$

Manuscript received October 23, 2007; revised December 10, 2007. This paper was recommended by Guest Editor H. Schmid.

The author is with SOTEKCO LLC, Baltimore, MD 21201 USA (e-mail: pps@ieee.org).

Digital Object Identifier 10.1109/TCSI.2008.919755

[1]Patent pending.

The Diophantine Frequency Synthesis (DFS) methodology, recently introduced in [8], [9], is one such multi loop approach. DFS is based on Diophantine equations [10] and can be implemented with two or more Integer-N PLLs that are driven by the same reference. The PLLs' output frequencies are added (or subtracted) to give the final output of the frequency synthesizer.

In DFS, a rational multiplication factor

$$a = \frac{n}{R_1 R_2 \cdots R_k}$$

with a typically very large numerator and denominator is decomposed into an algebraic sum of fractions of small positive integers

$$\frac{n_1}{R_1} \pm \frac{n_2}{R_2} \pm \cdots \pm \frac{n_k}{R_k}$$

using the theory of Diophantine equations [8], [10].

DFS allows for the realization of PLL-based architectures with both, very fine output frequency resolution, and, high PLL phase-comparator frequencies, simultaneously. High phase-comparator frequencies are easier to filter and allow for large PLL bandwidths and fast frequency hopping of the DFS synthesizer.

In a sense, DFS *distributes* the frequency resolution among the PLLs since the fractional frequency resolution at the output of the synthesizer equals to the product of the fractional frequency resolutions of the constituent PLLs.

Cascaded Diophantine Frequency Synthesis (CDFS), introduced in this paper, is a new systematic and mathematical methodology for developing and programming frequency synthesizer architectures based on Diophantine equations, offering all the advantages of DFS.

In addition, CDFS results in cascaded architectures in which the intermediate signals involved in the mixing stages have a minimal frequency range (within the general class of DFS synthesizers), typically half of that resulting from the DFS algorithm [8].

These characteristics of CFDS translate to more relaxed requirements on the frequency planning and design of mixers and filters as well as improved spurious-free dynamic range (SFDR) at the output. Moreover, the CFDS leads to modular and directly expandable circuit implementations with lower complexity than conventional multiloop architectures.

Finally, these properties give CDFS significant advantages over previous multiloop architectures using Diophantine equations that are based on heuristic algorithms and do not control the frequency ranges of the intermediate signals neither guaranty the existence of numerical solutions within a-priori determined ranges of pre-scaler and feedback dividers [11].

The paper presents the mathematical and algorithmic aspects of CDFS and is organized as follows: Section II introduces the notation used throughout the paper. Section III presents a

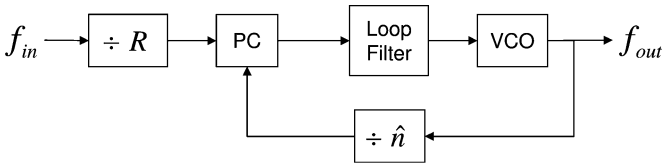

Fig. 1. Basic PLL.

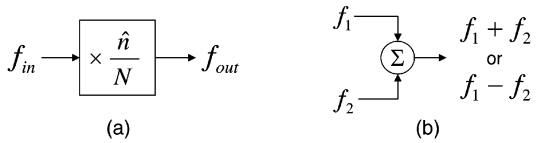

Fig. 2. (a) Simplified schematic of the basic PLL. (b) Mixer.

graphical example of frequency synthesis using properties of integer numbers. Comments on this basic example extend the questions to be answered by the general theory in the theory section. Section IV presents the general high-level CDFS architecture. Section V lays out the formal mathematical framework of CDFS. Section VI formulates the mathematical result into a practical algorithm for programming the CDFS synthesizers. Section VII discusses certain important parameters in frequency mixing in CDFS synthesizers and provides related bounds.

II. Notation

CDFS, like its predecessor DFS, uses two or more basic PLLs like the one in Fig. 1.[2] Throughout the paper, the *prescaler* divider, integer R, is assumed fixed. The *feedback divider*, integer $\hat{n}$, is the sum $\hat{n} = \bar{n} + n$ of a fixed integer $\bar{n}$ and an integer variable n which can take both *negative and positive values* within a predefined range. For all possible values of n, $\hat{n}$ is positive. The output frequency of the PLL is

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

Since the paper focuses on the mathematical principles and algorithms of CDFS and not on the technical details of the individual PLLs, Fig. 2(a) is used instead of Fig. 1 for convenience.

The mixing of two signals at frequencies f_1 and f_2 is denoted as in Fig. 2(b) where the outcome can be either $f_1 + f_2$ or $f_1 - f_2$. The context in the paper indicates whether the sum or the difference is considered.

Note that the term "mixer" here is used to represent the equivalent of the mixing along with pre- and post-filtering in a physical implementation where all unwanted intermodulation products (including $f_1 + f_2$ when the desirable output is $f_1 - f_2$, and vice versa) are appropriately removed.

Mixing of three or more signals has a similar interpretation. In practice, the order in which these signals are mixed is important in achieving a clean output spectrum. CDFS helps in doing so successfully.

III. CDFS: Motivation

CDFS can be illustrated in an intuitive way when it is applied to frequency synthesis architectures using two or more PLLs.

[2]"PC" may also be phase-frequency detector or similar block.

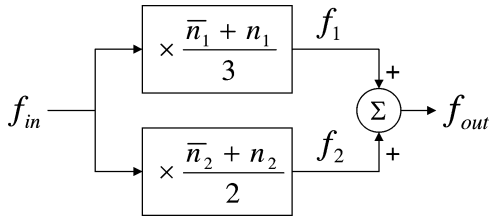

Fig. 3. Simple CDFS (DFS) scheme.

We begin with a single PLL, examine its performance limits and consider the improvements possible with a two-PLL system leading to a simple, yet revealing two PLL CDFS scheme. The CDFS is characterized followed by a generalization to the case of k-PLLs.

A. Inherent Limitations of PLLs

Frequency synthesis using a single PLL, like that in Fig. 1, implies frequency steps that are equal to the phase-comparator frequency f_{in}/R. The implication is that small frequency steps (via a large R and/or small f_{in}) necessitate a low phase-comparator frequency f_{in}/R resulting in a narrower loop bandwidth and slower frequency hopping in addition to a potential increase in spurs and noise [1], [2].

CDFS overcomes these constraints, simultaneously allowing for high phase-comparator frequencies and a very small output frequency step.

B. Example of A Basic 2-PLL CDFs Scheme

Now we ask the question: is it possible to achieve better resolution and/or higher loop bandwidths using two PLLs instead of one?

Consider the simple architecture in Fig. 3 involving two PLLs whose output frequencies are added. The prescalers of the PLLs are 3 and 2 (fixed). The feedback dividers are $\bar{n}_1 + n_1$ and $\bar{n}_2 + n_2$, respectively. Following the notation and assumptions of Section II we consider $\bar{n}_1$ and $\bar{n}_2$ to be fixed; their contribution to f_{out} is the fixed frequency part

$$\bar{f}_{\text{out}} = \left(\frac{\bar{n}_1}{3} + \frac{\bar{n}_2}{2}\right) f_{\text{in}}. \tag{1}$$

The variable parts of the feedback dividers are n_1 and n_2. It is reasonable to impose some bounds on them since this would be the case in every hardware implementation of the synthesizer. Let's assume that they can vary within the ranges

$$-3 \le n_1 \le 3 \qquad \text{and} \quad -2 \le n_2 \le 2 \tag{2}$$

respectively, i.e., the range of each feedback divider, $\bar{n}_i + n_i$, $i = 1, 2$, is twice the size of the corresponding prescaler 3, 2, respectively.[3] Finally, the output frequency is

$$f_{\text{out}} = \bar{f}_{\text{out}} + \left(\frac{n_1}{3} + \frac{n_2}{2}\right) f_{\text{in}}. \tag{3}$$

[3]This partially specifies the required frequency range of the voltage-controlled oscillators (VCOs).

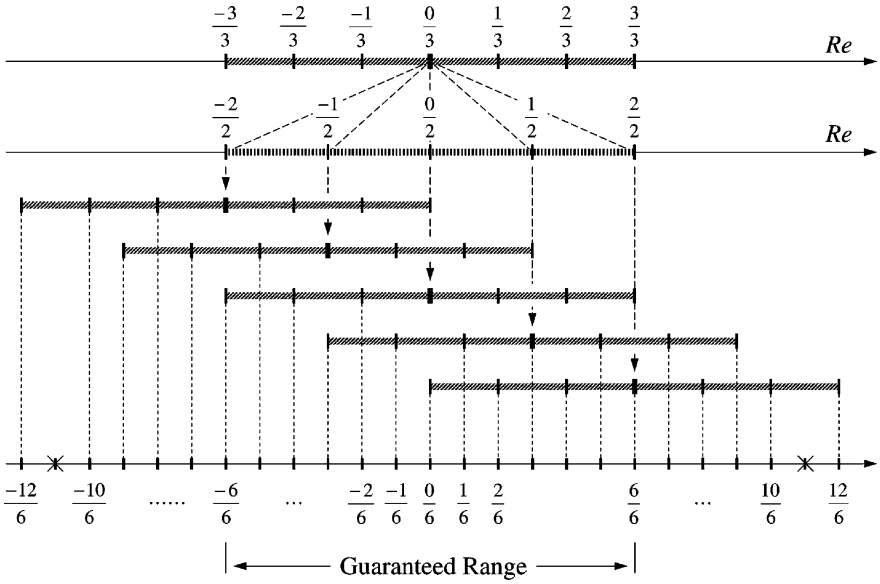

Fig. 4. Graphical representation of a simple CDFS / DFS scheme. All frequencies are normalized w.r.t. f_{in}.

Now let us focus on the variable part of f_{out} [second summand in the RHS of (3)] and to simplify things, let us assume [4] that $f_{\text{in}} = 1$ and $\bar{n}_1 = \bar{n}_2 = \bar{f}_{\text{out}} = 0$.

Given the bounds in (2), we can think of the first synthesizer as producing a set of 7 possible values of f_1, equally spaced by $f_{\text{in}}/3$, corresponding to the points of the top line in Fig. 4 and the second synthesizer (along with the mixer that adds the frequencies) as shifting this set by 5 possible offsets that are multiples of its resolution $f_{\text{in}}/2$. The end result is that f_{out} takes all values indicated on the bottom line of Fig. 4.

Comments: Following the above discussion and observing Fig. 4 we can point out the following.

1) The set of output frequencies contains the 13 values $-6/6, -5/6, \ldots, 6/6$ corresponding to frequency range
$$f_{\text{out}} \in [\bar{f}_{\text{out}} - f_{\text{in}}, \bar{f}_{\text{out}} + f_{\text{in}}]. \tag{4}$$
where here it is $\bar{f}_{\text{out}} = 0$ and $f_{\text{in}} = 1$.
2) Within the above output frequency range, the frequency step is *constant* and equal to
$$\text{frequency step} = \frac{f_{\text{in}}}{6} = \frac{f_{\text{in}}}{3 \cdot 2} \tag{5}$$
i.e., the choice of prescalers results in the much smaller frequency step, $f_{\text{in}}/6$, compared to those of the constituent PLLs, i.e., $f_{\text{in}}/3$ and $f_{\text{in}}/2$. In a sense, the resolution of f_{out} is "distributed" among the two PLLs.
3) If the mixer provided the frequency difference, i.e., if $f_{\text{out}} = f_1 - f_2$, properties 1 and 2 would still hold because the ranges of n_1 and n_2 are symmetric with respect to 0.
4) Combining the above and observing Fig. 4 we conclude that f_{out} can be expressed in the form
$$f_{\text{out}} = \bar{f}_{\text{out}} + \frac{n}{2 \cdot 3} f_{\text{in}} \tag{6}$$
where n takes, at least, the values $-6, -5, \ldots, 6$. In contrast, the output frequencies of the two constituent PLLs are
$$f_1 = \bar{f}_1 + \frac{n_1}{3} f_{\text{in}}$$
and
$$f_2 = \bar{f}_2 + \frac{n_2}{2} f_{\text{in}}.$$
5) Given the constraints in (2), achieving a particular value of f_{out} in (6), for some $n \in \{-6, -5, \ldots, 6\}$, may be possible in more than one ways, e.g., in Fig. 4 we see that
$$\frac{2}{3} = \frac{2}{3} + \frac{0}{2} = \frac{2}{2} + \frac{-1}{3}.$$
6) Fig. 4 shows that constraints (2) result in the 13 aforementioned values of f_{out} as well as others extending to $\pm 12/6$. Note however that the step is not constant within this extended $[-12/6, +12/6]$ range—the values $\pm 11/6$ are missing.
7) If instead of 3 and 2, the prescaler dividers where 3 and 6 respectively, one might intuitively expect a smaller frequency step of f_{out}, however, since
$$f_{\text{out}} = \bar{f}_{\text{out}} + \left(\frac{n_1}{3} + \frac{n_2}{6}\right) f_{\text{in}} \tag{7}$$
$$= \bar{f}_{\text{out}} + (2n_1 + n_2)\frac{f_{\text{in}}}{6} \tag{8}$$
and n_1 and n_2 are integers, the frequency step is not $f_{\text{in}}/(3 \cdot 6)$ but rather $f_{\text{in}}/6$ as before. □

We have pointed out the advantages and certain properties of the scheme in Fig. 3. In Section IV we will expand these results to address the general case of synthesizers with k PLLs.

The concepts we have discussed up to this point are also shared by the DFS methodology [8], a predecessor of CDFS. The need for a new theoretical development which led to CDFS and the advantages of CDFS over DFS are discussed next.

[4] In a practical case, it should be $\bar{n}_1 \geq 4$ and $\bar{n}_2 \geq 3$ so that for every n_1 and n_2 satisfying (2) it is $\bar{n}_1 + n_1 > 0$ and $\bar{n}_2 + n_2 > 0$.

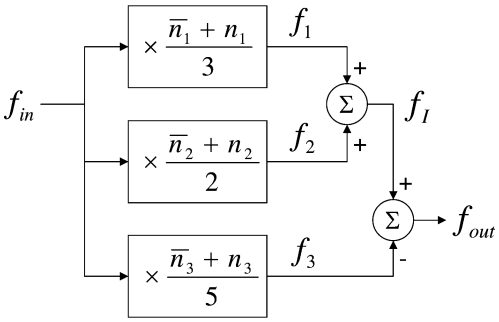

Fig. 5. Cascaded 3-PLL DFS scheme.

C. CDFs versus DFs

As it is shown in the following sections, CDFS methodology results in the same output frequency range and resolution as its predecessor DFS [8].

The significant advantage of CDFS over DFS becomes clear when we consider synthesizers with three or more PLLs. For instance, one more PLL can be added to the architecture in Fig. 3 resulting in the synthesizer shown in Fig. 5.

For the 3-PLL DFS scheme in Fig. 5, the DFS algorithm in [8] results in frequency ranges for f_1 and f_2 equal to $[\bar{f}_i - f_{\rm in}, \bar{f}_i + f_{\rm in}]$, $i = 1, 2$, respectively, but provides no information on the range of their sum (or difference) i.e., intermediate frequency f_I, even though the range of $f_{\rm out}$ is $[\bar{f}_{\rm out} - f_{\rm in}, \bar{f}_{\rm out} + f_{\rm in}]$. So, in general one has to assume the widest range for f_I, i.e.,

$$[\bar{f}_I - 2f_{\rm in}\,,\ \bar{f}_I + 2f_{\rm in}]. \tag{9}$$

In contrast, the CDFS algorithm introduced in Section VI along with the theory in Section V guarantee that f_I takes values within the range $[\bar{f}_I - f_{\rm in}, \bar{f}_I + f_{\rm in}]$ for every possible value of $f_{\rm out}$ in $[\bar{f}_{\rm out} - f_{\rm in}, \bar{f}_{\rm out} + f_{\rm in}]$. Therefore, CDFS results in a range of the intermediate frequency f_I that has half the length of (9).

Since in both DFS and CDFS the output frequency range and step are the same, having intermediate signals, like f_I, with the minimum possible frequency range is a significant advantage because it helps to achieve a clean output spectrum, simpler mixing stages, and easier frequency planning.

IV. General CDFS Scheme

Fig. 6 shows the general abstract high-level k-PLL DFS architecture [8]. The PLL parameters are integers R_i, n_i and $\bar{n}_i$, where $\bar{n}_i$ and R_i are fixed and such that $\bar{n}_i > R_i$ and n_i is a variable taking values within the range $-R_i \le n_i \le R_i$, for all $i = 1, 2, \ldots, k$.

All PLLs are driven by the same reference signal $f_{\rm in}$ and their output frequencies are added or subtracted (in any chosen but *fixed* pattern) resulting in the output frequency of the synthesizer

$$f_{\rm out} = \sum_{i=1}^{k} \alpha_i \left(\frac{\bar{n}_i}{R_i} + \frac{n_i}{R_i} \right) f_{\rm in} \tag{10}$$

where $\alpha_i \in \{-1, 1\}$, $i = 1, 2, \ldots, k$.

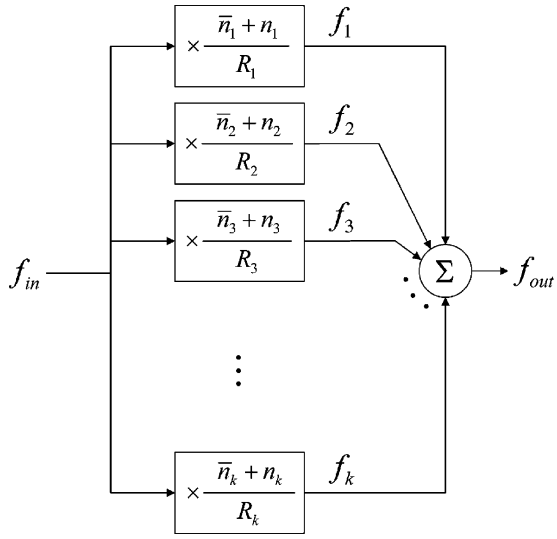

Fig. 6. Abstract high-level k-PLL DFS scheme.

The DFS methodology is valid for all combinations of signs of α_i s in (10) [8] and the same is true for CDFS as it is explained in the following sections. However, for presentation convenience it is easier to consider the case were all frequencies are added. Moreover, it is helpful to separate the variable from the fixed frequency component. With these in mind, we write

$$f_{\rm out} = \left(\frac{n_1}{R_1} + \frac{n_2}{R_2} + \cdots + \frac{n_k}{R_k} \right) f_{\rm in} + \bar{f}_{\rm out} \tag{11}$$

where $\bar{f}_{\rm out}$ is fixed and equal to

$$\bar{f}_{\rm out} \triangleq \left(\frac{\bar{n}_1}{R_1} + \frac{\bar{n}_2}{R_2} + \cdots + \frac{\bar{n}_k}{R_k} \right) f_{\rm in}. \tag{12}$$

Fig. 6 captures the desirable mathematical relationship between $f_{\rm in}$ and $f_{\rm out}$ without involving any information of the mixing process.

A major factor of the synthesizer's spectral quality is the order in which the k PLL output signals are mixed, i.e., the order in which they are added (or some of them are subtracted).

One way to realize these frequency additions (or subtractions) is to implement them in a cascade as shown in Fig. 7. Here the output frequency $f_{\rm out}$ is given by (10), or for simplicity, by (11) as well.

This abstract and general architecture in Fig. 7 along with its associated theory and programming algorithm presented in the following sections, is the CDFS.

To optimize the individual mixing stages in CDFS we need to derive the values and ranges of the intermediate frequencies

$$f_{I_j} = \bar{f}_{I_j} + \left(\frac{n_1}{R_1} + \frac{n_2}{R_2} + \cdots + \frac{n_j}{R_j} \right) f_{\rm in} \tag{13}$$

for $j = 1, 2, \ldots, k-1$ where $\bar{f}_{I_j}$ is fixed and equal to

$$\bar{f}_{I_j} = \left(\frac{\bar{n}_1}{R_1} + \frac{\bar{n}_2}{R_2} + \cdots + \frac{\bar{n}_j}{R_j} \right) f_{\rm in}. \tag{14}$$

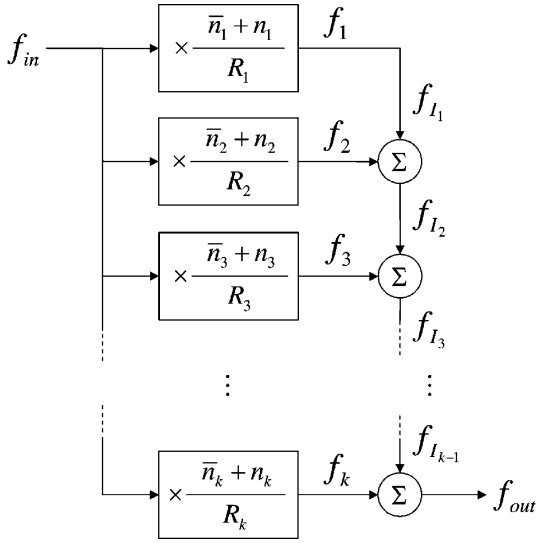

Fig. 7. k-PLL CDFS scheme.

For convenience we have defined $f_{I_1} \triangleq f_1$ as well. Again, without any loss of generality, it is more convenient to use expressions (13) and (14) instead of the more general

$$f_{I_j} = \sum_{i=1}^{j} \alpha_i \left(\frac{\bar{n}_i}{R_i} + \frac{n_i}{R_i} \right) f_{\text{in}}. \tag{15}$$

It is shown that the mathematical development in the following sections is valid for any combination of additions and subtractions. Change of the values of α_i s influences the central values of the intermediate and output frequencies but not their ranges (and steps).

Finally, mixing is an important part in any physical realization of CDFS and certain aspects of it are discussed in Section VII. Moreover, since CDFS is a special class of multiloop type architectures, proper signal filtering and isolation between the different stages is required to avoid parasitic coupling between the constituent PLLs and/or mixing stages that may degrade the quality of the output signal. To further avoid parasitic coupling between the PLLs it is preferable that the ratios R_i/R_j, $i, j = 1, 2, \ldots, k$, $i \neq j$, are sufficiently far from any ratio of small integers.

V. Mathematical Principles of CDFS

The discussion in the previous sections indicated that DFS and CDFS frequency synthesizers inherit their properties from those of the sum

$$\frac{n_1}{R_1} + \frac{n_2}{R_2} + \cdots + \frac{n_k}{R_k}.$$

Also, in CDFS architectures the partial sums

$$\frac{n_1}{R_1} + \frac{n_2}{R_2} + \cdots + \frac{n_j}{R_j}$$

where, $j = 2, 3, \ldots, k-1$ are involved in the frequency mixing stages and so their behavior influences the spectral quality of the output signal. The mathematical properties of these sums are studied in the following sections.

A. Main Theorem

The following Theorem is of fundamental importance for the CDFS methodology.

Theorem 5.1: Let $R_1, R_2, \ldots, R_k$ be pairwise relatively prime positive integers (i.e., no pair of them has common divider other than ± 1), then the statements A) and B) below are true.

A) For every integer n with $|n| \leq R_1 R_2 \cdots R_k$ we can find integers $n_1, n_2, \ldots, n_k$ with $|n_i| \leq R_i$, for all $i = 1, 2, \ldots, k$, satisfying Diophantine equation

$$\frac{n_1}{R_1} + \frac{n_2}{R_2} + \cdots + \frac{n_k}{R_k} = \frac{n}{R_1 R_2 \cdots R_k} \tag{16}$$

and inequalities (17) for $j = 2, 3, \ldots, k$

$$\left| \frac{n_1}{R_1} + \frac{n_2}{R_2} + \cdots + \frac{n_j}{R_j} \right| \leq 1. \tag{17}$$

B) There exist no constant $a < 1$ with the property that for every n with $|n| \leq R_1 R_2 \cdots R_k$ we can find a solution of (16) satisfying $|n_i| \leq a R_i$ for all $i = 1, 2, \ldots, k$. □

Comments: The following comments indicate the practical aspects of Theorem 5.1.

1) By appropriately adjusting the values of n_i, $i = 1, 2, \ldots, k$, and therefore adjusting the frequencies of the PLLs in Fig. 7, the output frequency, f_{out}, given by expression (11), can take all values within the range

$$[\bar{f}_{\text{out}} - f_{\text{in}} \quad , \quad \bar{f}_{\text{out}} + f_{\text{in}}] \tag{18}$$

(including $\bar{f}_{\text{out}}$) with *uniform* frequency step equal to

$$f_{\text{step}} = \frac{f_{\text{in}}}{R_1 R_2 \cdots R_k}. \tag{19}$$

Constant $\bar{f}_{\text{out}}$ is given by (12).

2) The statement of comment 1 above can be realized using only values of n_i s that satisfy

$$|n_i| \leq R_i, \qquad i = 1, 2, \ldots, k. \tag{20}$$

This is very important in practice because it specifies the ranges of the required values of the PLLs' feedback dividers $\bar{n}_i + n_i$.

Moreover, Part-B of the Theorem implies that the uniform (with respect to i) bounding (20) is optimal, i.e., there is no general class of solutions satisfying inequalities tighter than (20) and (17), for all $j = 1, 2, \ldots, k$.

3) For every n with $|n| \leq R_1 R_2 \cdots R_k$, the solution of Diophantine equation (16), whose existence is guaranteed by the Theorem 5.1, is such that the intermediate frequencies $f_{I_j}, j = 2, 3, \ldots, k-1$, (see Fig. 7) given by (13), satisfy the inequalities (inclusion)

$$f_{I_j} \in [\bar{f}_{I_j} - f_{\text{in}}, \ \bar{f}_{I_j} + f_{\text{in}}]. \tag{21}$$

These bounds are important in optimizing frequency mixing in CDFS scheme.

4) Constrains $|n_i| \leq R_i$, $i = 1, 2, \ldots, k$, in Theorem 5.1 are symmetric with respect to 0. Therefore, Theorem 5.1 would still be valid if we replace n_i by $-n_i$ for some of

the values of index i, i.e., if Diophantine equation (16) and inequalities (17) were replaced by

$$\sum_{i=1}^{k} \alpha_i \frac{n_i}{R_i} = \frac{n}{R_1 R_2 \cdots R_k}$$

and

$$\left| \sum_{i=1}^{j} \alpha_i \frac{n_i}{R_i} \right| \leq 1, \qquad j = 2, 3, \ldots, k$$

respectively. Where as before, $\alpha_i \in \{-1, 1\}$, $i = 1, 2, \ldots, k$ are arbitrarily chosen but fixed.

Application of this property of Theorem 5.1 to CDFS scheme in Fig. 7 implies that: If some of the frequencies $f_1, f_2, \ldots, f_k$ were subtracted from the general sum instead of being added, i.e., if we had used expressions (10) and (15) instead of (11) and (13), respectively, for a certain combination of $\alpha_i \in \{-1, 1\}$, $i = 1, 2, \ldots, k$, then comments 1, 2, and 3 above would still be valid. Moreover, the expressions of the ranges of $f_1, f_2, \ldots, f_k$, $f_{I_2}, f_{I_3}, \ldots, f_{I_{k-1}}$ and f_{out} as well as that of the step of f_{out} would be the same as before.

5) Theorem 5.1 guarantees the existence of solution within the specified bounds (17) and (20) but not the uniqueness of it. E.g., in Fig. 4 we see that equation

$$\frac{n_1}{3} + \frac{n_2}{2} = \frac{-4}{6}$$

has (at least) two solutions: $(n_1, n_2) = (-2, 0)$ and $(n_1, n_2) = (1, -2)$.

6) Theorem 5.1 guarantees the existence of solution within the bounds (17) and (20) when $|n| \leq R_1 R_2 \cdots R_k$. However, one may be able to find a solution satisfying these bounds for values of n such that $|n| > R_1 R_2 \cdots R_k$, e.g., in Fig. 4 we have

$$\frac{2}{3} + \frac{1}{2} = \frac{7}{6}.$$

The problem in general is that although we may be able to find solutions of (16) satisfying (17) and (20) for a value n_a of n with $R_1 R_2 \cdots R_k < n_a$, for example, we may not be able to do so for all integers n_b with $R_1 R_2 \cdots R_k < n_b < n_a$. So, even though the total range of n, and the range of f_{out}, may be larger than that given in Theorem 5.1, the step size $1/(R_1 R_2 \cdots R_k)$ is not guaranteed.

For example, the total range in Fig. 4 is $[-12/6, 12/6]$ but the points $\pm 11/6$ are missing. Especially for large $R_1, R_2, \ldots, R_k$, extension of the range beyond $\pm R_1 R_2 \cdots R_k$ is usually insignificant. □

B. Proof of Main Theorem

We define the set of positive integers $E_1, E_2, \ldots, E_k$ such that for $i = 1, 2, \ldots, k$

$$E_i = \prod_{\substack{j=1 \\ j \neq i}}^{k} R_j. \tag{22}$$

Note that the main equation, (16), which is repeated here for convenience

$$\frac{n_1}{R_1} + \frac{n_2}{R_2} + \cdots + \frac{n_k}{R_k} = \frac{n}{R_1 R_2 \cdots R_k} \tag{23}$$

is equivalent to (24) below, simply by multiplication by the product $E = R_1 R_2 \cdots R_k$

$$E_1 n_1 + E_2 n_2 + \cdots + E_k n_k = n. \tag{24}$$

Therefore, proving the existence of solution of (23) is equivalent to doing so for (24).

Lemma 5.1: Equation (24) has an integer solution $(n_1, n_2, \ldots, n_k)$ if and only if $\gcd(E_1, E_2, \ldots, E_k)$ divides n.

Proof: Let $\mathbb{Z}$ be the set of integers. From abstract Algebra we know that for any pair of integers a, b, it is $a\mathbb{Z} + b\mathbb{Z} = \gcd(a, b)\mathbb{Z}$. Using induction and noticing that $\gcd(\gcd(a, b), c) = \gcd(a, b, c)$, we get

$$\sum_{i=1}^{k} E_i \mathbb{Z} = \gcd(E_1, E_2, \ldots, E_k)\mathbb{Z}.$$

Therefore the values of n that can be generated by the LHS of (24) are exactly all integer multiples of $\gcd(E_1, E_2, \ldots, E_k)$. □

We also need the following lemma.

Lemma 5.2: The positive integers $R_1, R_2, \ldots, R_k$ are pairwise relatively prime (i.e., no pair of them has common divider other than ± 1) if and only if

$$\gcd(E_1, E_2, \ldots, E_k) = 1. \tag{25}$$

Proof: ($\Rightarrow$) Suppose $R_1, R_2, \ldots, R_k$ are pairwise relatively prime and $\gcd(E_1, E_2, \ldots, E_k) = m > 1$. Integer m has at least one prime divider $p > 1$. Since m divides E_1 it is also true that p divides E_1 and because $E_1 = R_2 R_3 \cdots R_k$ and p is prime, there exists $s \in \{2, \ldots, k\}$ such that p divides R_s. Since $R_1, R_2, \ldots, R_k$ are pairwise relatively prime, p does not divide R_i, $i \neq s$, therefore p does not divide E_s which contradicts our assumption.

($\Leftarrow$) Suppose $R_1, R_2, \ldots, R_k$ are not pairwise relatively prime, i.e., there exist $s, t \in \{1, \ldots, k\}$, $s \neq t$, such that $\gcd(R_s, R_t) = m > 1$. Then m divides all E_i, $i = 1, 2, \ldots, k$ since E_i has R_s or R_t as a factor. Therefore, $\gcd(E_1, E_2, \ldots, E_k) \geq m > 1$, which is a contradiction. □

Corollary 5.1: Diophantine equation (23) has a solution for every integer n if and only if $R_1, R_2, \ldots, R_k$ are pairwise relatively prime.

Proof: It follows from Lemmas 5.1 and 5.2, and the equivalence between (23) and (24). □

Note that if (23), or equivalently (24), has a solution, then it has infinitely many solutions. The details of this statement are given in Theorem 5.2 below (a proof can be found in [10]).

Theorem 5.2: If $\gcd(E_1, E_2, \ldots, E_k) = 1$ then there exists a fixed $k \times k$ integer matrix C with $\det(C) = 1$ such that for every integer n the complete set of solutions (row vectors of n_i's) of (24) is

$$\{(n, \xi_1, \xi_2, \ldots, \xi_{k-1})\, C \mid \xi_1, \xi_2, \ldots, \xi_{k-1} \in \mathbb{Z}\} \tag{26}$$

□

Dividing the vectors in the set (26) by $E = R_1R_2\cdots R_k$, we get the complete solution set of (23).

The case where $n = 0$ is of particular interest. The following lemma provides the exact form of the solution set (26) when $n = 0$.

Lemma 5.3: If $R_1, R_2, \ldots, R_k$ are pairwise relatively prime integers, then every solution of (27) is of the form $x_i = c_iR_i$, $i = 1, 2, \ldots, k$ where c_i s are integers satisfying the equation $c_1 + c_2 + \cdots + c_k = 0$

$$\frac{x_1}{R_1} + \frac{x_2}{R_2} + \cdots + \frac{x_k}{R_k} = 0. \tag{27}$$

Proof: Multiplying (27) by $R_1R_2\cdots R_k$ gives

$$E_1x_1 + E_2x_2 + \cdots + E_kx_k = 0. \tag{28}$$

By definition of E_j's, R_i divides E_j for all $j \neq i$. From (28), R_i also divides the product E_ix_i. Since $R_1, R_2, \ldots, R_k$ are pairwise relatively prime, $\gcd(E_i, R_i) = 1$ and so R_i must divide x_i. Therefore, $x_i = c_iR_i$ for some integers $c_1, c_2, \ldots, c_k$. Replacing them into (27) yields $c_1 + c_2 + \cdots + c_k = 0$. □

The above lemma provides the following Corollary that is central to the programming of cascaded Diophantine synthesizers. Its proof comes directly from the above.

Corollary 5.2: If $R_1, R_2, \ldots, R_k$ are pairwise relatively prime positive integers and $(x_1, x_2, \ldots, x_k)$ is a solution of

$$\frac{x_1}{R_1} + \frac{x_2}{R_2} + \cdots + \frac{x_k}{R_k} = \frac{1}{R_1R_2\cdots R_k} \tag{29}$$

then the general solution $(n_1, n_2, \ldots, n_k)$ of (23) is

$$(nx_1 + c_1R_1, nx_2 + c_2R_2, \ldots, nx_k + c_kR_k) \tag{30}$$

where c_is are integers satisfying $\sum_{i=1}^{k} c_i = 0$. □

Among these infinitely many solutions of (23) we are interested in the one(s) satisfying inequalities $-R_i \leq n_i \leq R_i$, $i = 1, 2, \ldots, k$, and (17).

Lemma 5.4: Let R_1, R_2 be relatively prime positive integers, then, for every integer n, in the range $-R_1R_2 \leq n \leq R_1R_2$, we can find a solution (n_1, n_2) of

$$\frac{n_1}{R_1} + \frac{n_2}{R_2} = \frac{n}{R_1R_2} \tag{31}$$

such that for $i = 1, 2$

$$-R_i \leq n_i \leq R_i. \tag{32}$$

Proof: If $n = \pm R_1R_2$ then $(n_1, n_2) = (\pm R_1, 0)$, respectively, is an appropriate solution. Now consider the case $-R_1R_2 < n < R_1R_2$. From Corollary 5.1 we know that a solution (x_1, x_2) of (31) exists, i.e.,

$$\frac{x_1}{R_1} + \frac{x_2}{R_2} = \frac{n}{R_1R_2}. \tag{33}$$

For $i = 1, 2$ set $z_i = x_i \bmod R_i$ and $\mu_i = x_i \operatorname{div} R_i$, then, $x_i = z_i + \mu_iR_i$. Substituting the last expressions into (33) we get

$$\frac{z_1}{R_1} + \frac{z_2}{R_2} + \mu_1 + \mu_2 = \frac{n}{R_1R_2}. \tag{34}$$

By the assumption on n and the definitions of z_is it is

$$-1 < \frac{n}{R_1R_2} < 1$$

and

$$0 \leq \frac{z_1}{R_1} + \frac{z_2}{R_2} < 2$$

which along with (34) and the fact that μ_1 and μ_2 are integers, imply that $\mu_1 + \mu_2 \in \{-2, -1, 0\}$. Now we set

$$(n_1, n_2) = \begin{cases} (z_1 - R_1, z_2 - R_2), & \text{if } \mu_1 + \mu_2 = -2 \\ (z_1 - R_1, z_2), & \text{if } \mu_1 + \mu_2 = -1 \\ (z_1, z_2), & \text{if } \mu_1 + \mu_2 = 0 \end{cases} \tag{35}$$

and we note that in all cases, (n_1, n_2) is a solution of (31), and since by the definition of z_i's it is $0 \leq z_i < R_i$, $i = 1, 2$, we also have that $-R_i \leq n_i \leq R_i$, $i = 1, 2$. □

Definition 5.1: Let (R_a, R_b) be a pair of relatively prime positive integers and let $\mathcal{S}_{(R_a,R_b)}$ be the set

$$\mathcal{S}_{(R_a,R_b)} = \left\{ (x_a, x_b) \middle| \frac{x_a}{R_a} + \frac{x_b}{R_b} = \frac{n}{R_aR_b}, |n| \leq R_aR_b \right\}.$$

We define the following function:

$$f_{(R_a,R_b)} : \mathcal{S}_{(R_a,R_b)} \to \{-R_a, \ldots, R_a\} \times \{-R_b, \ldots, R_b\} \tag{36}$$

based on the procedure in the proof of Lemma 5.4, that converts an arbitrary solution (x_1, x_2) of (33) to a solution (n_1, n_2), given by (35), that satisfies inequalities (32), i.e.,

$$f_{(R_1,R_2)}(x_1, x_2) = (n_1, n_2).$$

Note that $\{f_{(R_a,R_b)}\}$ is a family of functions parameterized on the pair of relatively prime integers (R_a, R_b). □

Now we use the results above to prove Theorem 5.1 in Section V-A.

Proof of Theorem 5.1: Let $R_1, R_2, \ldots, R_k$ be pairwise relatively prime positive integers. We prove Part-A first, i.e., that for every integer n satisfying $|n| \leq R_1R_2\cdots R_k$ we can find integers $n_1, n_2, \ldots, n_k$ with $|n_i| \leq R_i$, $i = 1, 2, \ldots, k$, satisfying (37) and (38) below for all $j = 2, 3, \ldots, k$

$$\frac{n_1}{R_1} + \frac{n_2}{R_2} + \cdots + \frac{n_k}{R_k} = \frac{n}{R_1R_2\cdots R_k} \tag{37}$$

$$\left| \frac{n_1}{R_1} + \frac{n_2}{R_2} + \cdots + \frac{n_j}{R_j} \right| \leq 1. \tag{38}$$

Lemma 5.4 proves this for $k = 2$. We use induction and assume the statement above is true for $k = m$ to prove it is also true for $k = m + 1$.

Consider equation (39) where ν is an integer such that $|\nu| \leq R_1R_2\cdots R_{m+1}$

$$\frac{\mu}{R_1R_2\cdots R_m} + \frac{n_{m+1}}{R_{m+1}} = \frac{\nu}{R_1R_2\cdots R_{m+1}}. \tag{39}$$

Since $R_1R_2\cdots R_m$ and R_{m+1} are pairwise relatively prime, from Lemma 5.4 we know that there exists a solution (μ, n_{m+1}) such that $|\mu| \leq R_1R_2\cdots R_m$ and $|n_{m+1}| \leq R_{m+1}$. By induction's hypothesis we also have that there exist $n_1, n_2, \ldots, n_m$ with $|n_i| \leq R_i$, for all $i = 1, 2, \ldots, m$, for which

$$\frac{n_1}{R_1} + \frac{n_2}{R_2} + \cdots + \frac{n_m}{R_m} = \frac{\mu}{R_1R_2\cdots R_m} \tag{40}$$

$$\left|\frac{n_1}{R_1} + \frac{n_2}{R_2} + \cdots + \frac{n_j}{R_j}\right| \leq 1 \tag{41}$$

for all $j = 2, 3, \ldots, m$. Combining (39), (40), and (41) with the bounds on μ, n_{m+1} and ν we prove the induction step and conclude the proof of Part-A of Theorem 5.1.

The proof of Part-B of Theorem 5.1 comes from Corollary 5.2. Specifically, every solution $(n_1, n_2, \ldots, n_k)$ of (37) for $n = R_1R_2\cdots R_k$ is of the form

$$((c_1+1)R_1, c_2R_2, \ldots, c_kR_k)$$

where integers $c_1, c_2, \ldots, c_k$ satisfy $c_1 + c_2 + \ldots + c_k = 0$. Therefore, it must be $|n_i| \geq N_i$ for at least one value of index i. □

VI. Algorithms

Theorem 5.1 in Section V-A shows that there exists a solution of (16) satisfying inequalities (17) and (20). Such solutions are important in practice because they result in *a priori* known and tight ranges of values of the feedback dividers in CDFS schemes in Fig. 7.

If only a single and fixed output frequency is desirable, then Diophantine equation (16) must be solved once for a particular value of n. If many output frequencies may be needed, then (16) must be solved for all corresponding values of n. Solving the equation requires some computational effort and in most cases, long-integer algebraic manipulation.

To avoid this computational complexity, one could consider storing the solutions, corresponding to all possible values n, 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.

Alternatively, one can use the solutions of a set of particular equations to generate, in a simple way, a solution of (16), as specified in Theorem 5.1, for every value of n. The CDFS algorithm achieving this is presented below. It can be used in real-time or off-line to derive the parameters of the synthesizer.

CDFS Algorithm

- STEP 1: For $i = 1, 2, \ldots, k-1$ derive and store a solution (z_{k-i}, w_{k-i+1}) of the Diophantine equation

$$\frac{z_{k-i}}{R_1R_2\cdots R_{k-i}} + \frac{w_{k-i+1}}{R_{k-i+1}} = \frac{1}{R_1R_2\cdots R_{k-i+1}}. \tag{42}$$

  For numerical stability it is preferable that the derived solutions involve absolutely small numbers.

- STEP 2: Set $x_k = n$ and derive sequentially the $k-1$ vectors (x_{k-i}, n_{k-i+1}), $i = 1, 2, \ldots, k-1$, using

$$(x_{k-i}, n_{k-i+1}) = f_{(\prod_{\ell=1}^{k-i} R_\ell,\, R_{k-i+1})}(x_{k-i+1}z_{k-i},\, x_{k-i+1}w_{k-i+1}). \tag{43}$$

- STEP 3: Set $n_1 = x_1$.

Vector $(n_1, n_2, \ldots, n_k)$ is a solution of Diophantine equation (16), having the properties specified in Theorem 5.1. The proof of this statement comes directly from the properties of the class of functions $f_{(R_a,R_b)}$ and use of induction.

It is instructive to discuss the case $k = 4$ step-by-step in detail—this is done in Section VI-A.

Comments: Some properties of the algorithm are summarized below.

1) In Step 1, it is required that we solve $k-1$ versions of the linear Diophantine equation (42) of two variables. Note that $R_1, R_2, \ldots, R_k$ are pairwise prime, by assumption, and so $\gcd(R_1R_2\cdots R_{k-i}, R_{k-i+1}) = 1$. Therefore, the existence of the solutions is guaranteed by Theorem 5.1. A solution can be found using the Euclidean algorithm [10]. For most practical purposes this can be done using the Euclidean algorithm executed by the "gcd" command in MATLAB. Specifically, if m_1, m_2 are two relatively prime integers, then executing "$[g, x_1, x_2] = \gcd(m_1, m_2)$" in MATLAB returns a vector (g, x_1, x_2) where (x_1, x_2) is a solution $m_1x_1 + m_2x_2 = 1$ and $g = \gcd(m_1, m_2) = 1$. Note however that the product $R_1R_2\cdots R_{k-i}$ in (42) may have a large value. To avoid numerical errors due to overflows in MATLAB it is advisable to use the Variable Precision Arithmetic (VPA) command and specify the size of the largest integers that can be handled.
2) Step 1 is executed only once and the derived pairs (z_1, w_2), $(z_2, w_3), \ldots, (z_{k-1}, w_k)$ are stored to be used in Step(s) 2 in real-time or off-line operation. The storage requirements are minimal.
3) To derive an appropriate solution (according to Theorem 5.1) for a certain value of n, and program the frequency synthesizer in real time, we use the pairs (z_{k-i}, w_{k-i+1}) derived and stored in Step 1 and execute Step 2 and Step 3. Note that a different "version" of function "f" is used in each of the iterations of Step 2.
4) The arithmetic operations required in Step 2 are: integer addition, subtraction, multiplication and modulo. The size of the integers involved depends on the stored pairs (z_{k-i}, w_{k-i+1}) and implicitly on $R_1, R_2, \ldots R_k$, and it can be determined in advance off-line.
5) For discrete-component CDFS synthesizer designs the CDFS algorithm can be programmed into a Field-Programmable Gate Array (FPGA) or a micro-controller. If the algorithm is used in real-time, the memory requirements are negligible; if it is used off-line and the programming parameters are stored, a total of

$$\cong (2R_1R_2\cdots R_k) \times k \times \log_2(\max_i\{R_i\})$$

bits are required. □

A. Example: Application of CDFs Algorithm for $k = 4$

To illustrate the application of the CDFS algorithm, the case $k = 4$ is considered. The three steps of the algorithm follow in explicit form.

Let n be an integer such that $|n| \leq R_1R_2R_3R_4$.

- STEP 1: Derive the solutions (z_3, w_4), (z_2, w_3) and (z_1, w_2) of the following Diophantine equations using the Euclidean algorithm:

$$\frac{z_3}{R_1R_2R_3} + \frac{w_4}{R_4} = \frac{1}{R_1R_2R_3R_4} \tag{44}$$

$$\frac{z_2}{R_1R_2} + \frac{w_3}{R_3} = \frac{1}{R_1R_2R_3} \tag{45}$$

$$\frac{z_1}{R_1} + \frac{w_2}{R_2} = \frac{1}{R_1R_2}. \tag{46}$$

- STEP 2: Set $x_4 = n$ and derive sequentially vectors (x_3, n_4), (x_2, n_3) and (x_1, n_2) using "three versions" of function f:

$$(x_3, n_4) = f_{(R_1R_2R_3,\, R_4)}(nz_3,\, nw_4) \tag{47}$$

$$(x_2, n_3) = f_{(R_1R_2,\, R_3)}(x_3z_2,\, x_3w_3) \tag{48}$$

$$(x_1, n_2) = f_{(R_1,\, R_2)}(x_2z_1,\, x_2w_2). \tag{49}$$

- STEP 3: Set $n_1 = x_1$. □

Comments: Certain points of interest are discussed.

1) In Step 1 equations (44)–(46) correspond to $i = 1, 2$ and 3, respectively, in the CDFS algorithm. Similarly for equations (47)–(49) in Step 2.
2) Equation (44) along with (47) and the definition of function $f_{(R_1R_2R_3,\, R_4)}$ imply that

$$\frac{x_3}{R_1R_2R_3} + \frac{n_4}{R_4} = \frac{n}{R_1R_2R_3R_4} \tag{50}$$

similarly from (45), (48), (46) and (49), we get that

$$\frac{x_2}{R_1R_2} + \frac{n_3}{R_3} = \frac{x_3}{R_1R_2R_3} \tag{51}$$

$$\frac{x_1}{R_1} + \frac{n_2}{R_2} = \frac{x_2}{R_1R_2}. \tag{52}$$

From Step 3 and (52), we also get

$$\frac{n_1}{R_1} + \frac{n_2}{R_2} = \frac{x_2}{R_1R_2}. \tag{53}$$

By adding (51) and (53), we get

$$\frac{n_1}{R_1} + \frac{n_2}{R_2} + \frac{n_3}{R_3} = \frac{x_3}{R_1R_2R_3}. \tag{54}$$

Finally, adding (50) to (54) gives

$$\frac{n_1}{R_1} + \frac{n_2}{R_2} + \frac{n_3}{R_3} + \frac{n_4}{R_4} = \frac{n}{R_1R_2R_3R_4}. \tag{55}$$

3) Now note that integers n_1, n_2, n_3 and n_4 form a solution of (55) and satisfy (53) and (54) as well. Moreover, since it is $|n| \leq R_1R_2R_3R_4$ and given the properties of functions f, we also get that

$$\left|\frac{n_1}{R_1} + \frac{n_2}{R_2}\right| = \left|\frac{x_2}{R_1R_2}\right| \leq 1$$

and

$$\left|\frac{n_1}{R_1} + \frac{n_2}{R_2} + \frac{n_3}{R_3}\right| = \left|\frac{x_3}{R_1R_2R_3}\right| \leq 1.$$

Therefore solution (n_1, n_2, n_3, n_4) satisfies all the desirable properties stated in Theorem 5.1. □

VII. CDFS Frequency Mixing Aspects

Mixing is an important part of frequency synthesis and one should examine carefully all aspects of it in order to achieve clean output signal and desirable spurious-free dynamic range (SFDR) over the whole range of synthesized frequencies.

For extensive discussion on frequency mixing technics and issues there are many excellent references including [3] and [12]–[14].

The intention of this section is to link the mathematical development presented in Section V to some critical parameters of the mixing processes in CDFS schemes.

A major parameter of the mixing process of two signals at frequencies f_1 and f_2, that partially defines the spurs' level, is the ratio f_2/f_1 and its range of values. The importance of this ratio is much more profound in wide-band frequency synthesizers and RF systems.

A. CDFs: Frequency Ratios At the Mixers

Consider the abstract CDFS scheme in Fig. 7 with intermediate frequencies f_{I_j}, $j = 2, 3, \ldots, k-1$. For convenience we also set $f_{I_1} \triangleq f_1$ and $f_{I_k} \triangleq f_{\text{out}}$.

From Theorem 5.1 we have bounds (18) and (21), as well as $-R_i \leq n_i \leq R_i$ for $i = 1, 2, \ldots, k$. Summarizing we have that for all $j = 1, 2, \ldots, k$

$$\bar{f}_j - f_{\text{in}} \leq f_j \leq \bar{f}_j + f_{\text{in}} \tag{56}$$

$$\bar{f}_{I_j} - f_{\text{in}} \leq f_{I_j} \leq \bar{f}_{I_j} + f_{\text{in}} \tag{57}$$

$$\bar{f}_{\text{out}} - f_{\text{in}} \leq f_{\text{out}} \leq \bar{f}_{\text{out}} + f_{\text{in}} \tag{58}$$

where

$$\bar{f}_j = \frac{\bar{n}_j}{R_j} f_{\text{in}}$$

and

$$\bar{f}_{I_j} = \left(\sum_{i=1}^{j} \alpha_i \frac{\bar{n}_i}{R_i}\right) f_{\text{in}}$$

and $\alpha_i \in \{-1, 1\}$, $i = 1, 2, \ldots, k$ are defined according to the desirable summation/subtraction pattern in the mixers as in Section IV. In the jth mixing stage in Fig. 7 we mix f_{I_j} with f_{j+1}, for $j = 1, 2, \ldots, k-1$.

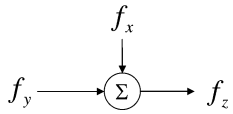

Fig. 8. Generic Mixing Notation: $f_x \geq f_y$, $f_{\text{out}} = f_x \pm f_y$.

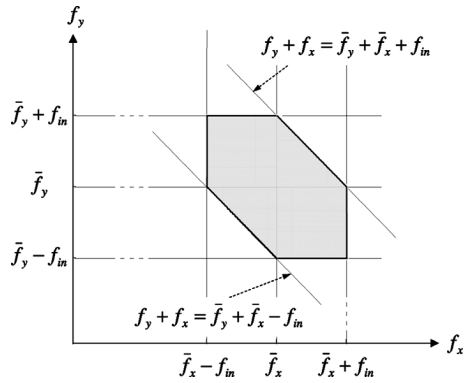

Fig. 9. Graphical representation of the inequality constraints (59)–(61) and the feasible pairs (f_x, f_y) when $f_z = f_x + f_y$.

Since in all cases the ranges of the frequencies are similar, it is convenient to use symbols f_x and f_y to represent any mixing pair, and, f_z to represent the outcome of the mixing as in Fig. 8. So it is either $f_z = f_x + f_y$ or $f_z = f_x - f_y$ and inequalities (56)–(58) translate to

$$\bar{f}_x - f_{\text{in}} \leq f_x \leq \bar{f}_x + f_{\text{in}} \tag{59}$$

$$\bar{f}_y - f_{\text{in}} \leq f_y \leq \bar{f}_y + f_{\text{in}} \tag{60}$$

$$\bar{f}_z - f_{\text{in}} \leq f_z \leq \bar{f}_z + f_{\text{in}}. \tag{61}$$

Moreover, without any loss of generality we can also assume that[5]

$$f_x \geq f_y. \tag{62}$$

1) Case 1: Frequency Addition: Suppose that $f_z = f_x + f_y$. The three constraints (59)–(61) are shown graphically in Fig. 9 and all feasible pairs (f_x, f_y) lie within the gray polygon.

We are interested in the range of values of the ratio $\rho = f_y/f_x$. The geometry of the boundary, see Fig. 10, implies that ρ is maximized at the top-left vertex and minimized at the bottom-right one. So we have

$$\rho_M = \frac{\bar{f}_y + f_{\text{in}}}{\bar{f}_x - f_{\text{in}}}$$

and

$$\rho_m = \frac{\bar{f}_y - f_{\text{in}}}{\bar{f}_x + f_{\text{in}}}.$$

2) Case 2: Frequency Subtraction: Now consider the case $f_z = f_x - f_y$. It is assumed that $f_x \geq f_y$ for all possible values of the two frequencies. The three constraints (59)–(61) are shown graphically in Fig. 11 and all feasible pairs (f_x, f_y) lie within the gray polygon which is rotated by 90° with respect to that in Fig. 9.

[5]This assumption is important for the graphical representation in Fig. 12 but not for Figs. 9–11 where it is ignored for convenience.

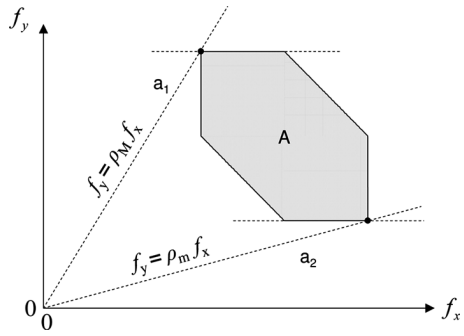

Fig. 10. Maximum and minimum ratios f_x/f_y when $f_z = f_x + f_y$.

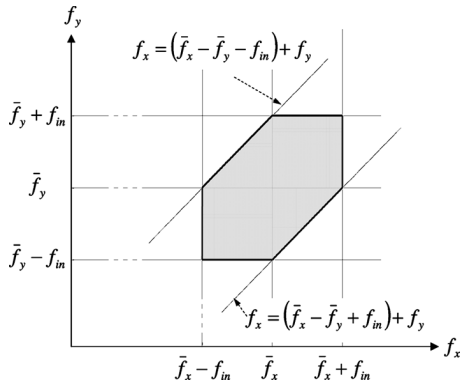

Fig. 11. Graphical representation of the inequality constraints (59)–(61) and the feasible pairs (f_x, f_y) when $f_z = f_x - f_y$.

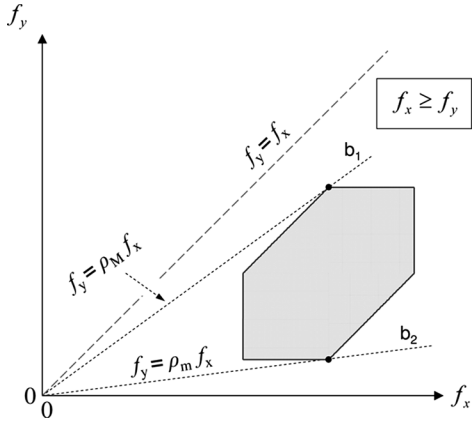

Fig. 12. Maximum and minimum ratios f_x/f_y when $f_z = f_x - f_y$.

Since $f_x \geq f_y$ for all feasible pairs (f_x, f_y), the polygon is entirely below the first diagonal, line $f_x = f_y$. The maximum and minimum values of the ratio $\rho = f_y/f_x$ are also reached at the top-left and the bottom-right vertices respectively, as shown in Fig. 12. Therefore

$$\rho_M = \frac{\bar{f}_y + f_{\text{in}}}{\bar{f}_x}$$

and

$$\rho_m = \frac{\bar{f}_y - f_{\text{in}}}{\bar{f}_x}.$$

VIII. Conclusion

The CDFS methodology has been introduced. It is based on Diophantine equations and employs two or more PLLs to achieve frequency resolution arbitrarily finer than that of the constituent PLLs without compromising neither hopping speed nor loop bandwidth.

CDFS results in controlled, narrow frequency ranges of all intermediate signals in the frequency mixing steps allowing for improved spectral purity and design convenience. CDFS provides new algorithms for programming the constituent PLLs accordingly. CDFS leads to fine frequency step, fast frequency hopping architectures with potentially very low spurs, especially in the vicinity of the carrier.

The mathematical principles and algorithmic aspects of CDFS as well as the impact of CDFS to the frequency mixing stages have been discussed.

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] U. L. Rohde, *Microwave and Wireless Synthesizers: Theory and Design*, 1st ed. Singapore: Wiley-Interscience, 1997.

[4] J. A. Crawford, *Frequency Synthesizer Design Handbook*. Natick, MA: Artech House, Jul. 1994.

[5] C. S. Vaucher, *Architectures for RF Frequency Synthesizers*. New York: Springer, 2002.

[6] R. E. Best, *Phase-Locked Loops: Design, Simulation, and Applications*, 5th ed. New York: McGraw-Hill Professional, 2003.

[7] V. F. Kroupa, *Direct Digital Frequency Synthesizers*. New York: Wiley-IEEE Press, 1998.

[8] P. P. Sotiriadis, "Diophantine frequency synthesis," *IEEE Trans. Ultrason., Ferroelectr., Freq. Contr.*, vol. 53, no. 11, pp. 1988–1998, Nov. 2006.

[9] P. P. Sotiriadis, "Diophantine frequency synthesis: The mathematical principles," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2007.

[10] D. E. Flath, *Introduction to Number Theory*. New York: Wiley, 1989.

[11] W. G. Wilke, "Diophantine Synthesizer," U.S. Patent 5267182, 1993.

[12] S. A. Maas, *Microwave Mixers*, 2nd ed. Natick, MA: Artech House, 1993.

[13] W. F. Egan, *Practical RF System Design*. New York: Wiley-IEEE Press, 2003.

[14] G. D. Vendelin, A. M. Pavio, and L. Ulrich, *Rohde Microwave Circuit Design Using Linear and Nonlinear Techniques*, 2nd ed. Singapore: Wiley-Interscience, 2005.

Paul Peter Sotiriadis (S'99–M'02) received the diploma in electrical and computer engineering from the National Technical University of Athens, Athens, Greece, in 1994, the M.S. degree in electrical engineering from Stanford University, Stanford, CA, in 1996, and the Ph.D. degree in electrical engineering and computer science from the Massachusetts Institute of Technology, Cambridge, in 2002.

He joined Johns Hopkins University in 2002 as Assistant Professor of Electrical and Computer Engineering. His research interests include design, optimization, and mathematical modeling of analog and mixed-signal circuits, RF and microwave circuits, advanced frequency synthesis, timekeeping systems, biomedical instrumentation, interconnect networks in deep-sub-micron and nano-technologies.

Dr. Sotiriadis serves as an Associate Editor of the IEEE Transactions on Circuits and Systems—II: Express Briefs. He has been a member of the technical committees of several conferences. He regularly reviews for many IEEE Transactions and conferences. He also regularly serves on proposal review panels at the National Science Foundation.