Principles of Cascaded Diophantine Frequency Synthesis

Paul Peter Sotiriadis

pps@ieee.org

***Abstract*— Cascaded Diophantine Frequency Synthesis[1] (CDFS) is a new systematic methodology for developing and programming modular multi-loop frequency synthesizers with high frequency resolution, fast frequency hopping and potentially very low spurs, especially near-in. CDFS results in significantly reduced frequency ranges of the intermediate signals in the frequency mixing stages compared to the predecessor, Diophantine Frequency Synthesis methodology. This simplifies the design and frequency planing for the synthesizer and allows for improved spectral purity of the output signal.**

I. Introduction

Cascaded Diophantine Frequency Synthesis (CDFS) is based on Diophantine equations [1] and shares the same mathematical principles with its predecessor, Diophantine Frequency Synthesis (DFS) methodology, that was introduced in the IEEE Frequency Control Symposium of 2006, [2].

Both methods employ two or more Frequency Synthesis Blocks (FSB), like Phase Locked Loops (PLL), Direct Digital Synthesizers (DDS) or others[2] [3], which are driven by the same reference frequency signal and whose output frequencies are added or subtracted to give the output frequency of the DFS / CDFS synthesizer.

Both methods use only exactly-periodic signals, without employing any dithering, interpolation, pulse removal or other approximately-periodic waveform that may corrupt the near-in spectrum[3], and, they result in high-level architectures whose output fractional-frequency resolution is equal to the product of the FSB's fractional-frequency resolutions. This allows for the output frequency step to be made (arbitrarily) small while using a relatively high frequency reference for the FSBs.

CDFS has all desirable properties of DFS and in addition it offers the following significant advantage: it minimizes the frequency ranges of all intermediate signals involved in frequency mixing which allows for improved spectral purity and lower design complexity compared to DFS, [4]-[5].

[1]Patent Pending

[2]Almost any frequency synthesizer can be used as a constituent block of DFS or CDFS synthesizers; in general one would prefer simple(r) structures like Integer-N PLLs.

[3]However, one may choose FSBs that do so.

It is instructive to summarize DFS and discuss the frequency properties of the intermediate signals involved in the frequency mixers, first, in order to introduce and motivate CDFS.

II. Notation

The Frequency Synthesis Blocks (FSB)s used to compose a CDFS or DFS synthesizer are shown in abstract form in Figure 1a as frequency multipliers by $\hat{n}/R$.

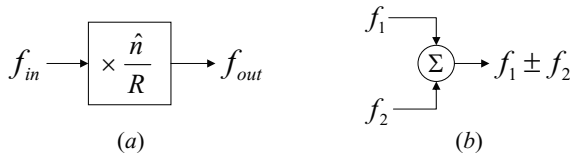

Fig. 1. (*a*) High-level notation for Frequency Synthesis Blocks (FSB), (*b*) Frequency mixer notation - the output is either $f_1 + f_2$ or $f_1 - f_2$.

Both the frequency multiplier $\hat{n}$ and the frequency divider R are assumed to be *positive*. Moreover, since most of the possible FSB have output frequency range extending from a non-zero minimum frequency up to a maximum frequency[4] it is useful to decompose the frequency multiplier into the sum $\hat{n} = \bar{n} + n$, where $\bar{n}$ is fixed and greater than R, and, n is an integer variable ranging from $-R$ to $+R$. The output frequency is

$$f_{out} = \frac{\bar{n} + n}{R} f_{in}. \tag{1}$$

An additional, fixed, frequency divider, Q, may also be be present in (1), i.e. $f_{out} = \frac{\bar{n}+n}{Q\,R} f_{in}$. Moreover, the range of n can also be expanded in multiples of R.

The mixing of two signals at frequencies f_1 and f_2 is denoted as in Figure 1b where the outcome can be either $f_1 + f_2$ or $f_1 - f_2$ (fixed choice). The context in the paper indicates whether the sum or the difference is considered. The mixing process here includes pre- and post-filtering to remove all unwanted intermodulation products (including $f_1 + f_2$ when the desirable output is $f_1 - f_2$, and vice versa). Mixing of three or more signals has a similar interpretation.

978-1-4244-1795-7/08/$25.00 ©2008 IEEE

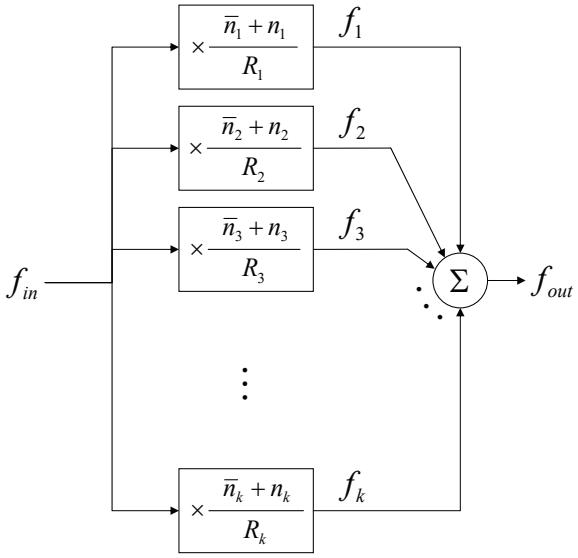

Fig. 2. Abstract high-level k-PLL DFS scheme.

III. Summary of DFS' Properties

DFS uses two or more basic FSBs, like that in Fig. 1, as shown in Fig. 2 depicting the general abstract high-level k-FSB DFS architecture. The FSBs, have parameters, R_i, n_i and $\bar{n}_i$, $i = 1, 2, \ldots, k$, and are driven by the same input reference f_{in}. Their output frequencies are added, or subtracted, (in any - but fixed pattern) to give the output frequency f_{out} of the synthesizer

$$f_{out} = \bar{f}_{out} + \left(\sum_{i=1}^{k} \alpha_i \frac{n_i}{R_i} \right) f_{in} \tag{2}$$

where $\alpha_i \in \{-1, 1\}$, $i = 1, 2, \ldots, k$, are the frequency - weighting coefficients, whose values correspond to whether a frequency is added or subtracted from the general sum in the mixing process, and,

$$\bar{f}_{out} = \left(\sum_{i=1}^{k} \alpha_i \frac{\bar{n}_i}{R_i} \right) f_{in} \tag{3}$$

is the central (fixed) value of the output frequency f_{out}.

Now following the DFS methodology [6] the FSBs are designed so that $\bar{n}_i$'s and R_i's are fixed, $\bar{n}_i > R$, and n_i can take any integer value from $-R$ to $+R$. Therefore, for the output frequencies of the FSBs we have

$$f_i \in \left[\bar{f}_i - f_{in} \ , \ \bar{f}_i + f_{in} \right] \tag{4}$$

where the central output frequencies of the FSBs are

$$\bar{f}_i = \frac{\bar{n}_i}{R_i} f_{in}. \tag{5}$$

[4]Note that DDS and some multiloop synthesizers, used as FSBs, may have zero minimum output frequency.

Moreover, the positive integers $R_1, R_2, \ldots, R_k$ are chosen to be *pairwise relatively prime*. Then DFS guarantees that by programming the values of n_is using the DFS algorithm [6], f_{out}, can take (all) values within the range

$$\left[\bar{f}_{out} - f_{in} \ , \ \bar{f}_{out} + f_{in} \right] \tag{6}$$

with *uniform* frequency step equal to

$$f_{step} = \frac{f_{in}}{R_1 R_2 \cdots R_k}. \tag{7}$$

where $\bar{f}_{out}$ is given by (3). More specifically, f_{out} can take any of the values

$$f_{out} = \bar{f}_{out} + \frac{n}{R_1 R_2 \cdots R_k} f_{in}, \tag{8}$$

where

$$n = -R_1 R_2 \cdots R_k, \ \ldots \ , R_1 R_2 \cdots R_k, \tag{9}$$

by programming the values of n_i's within their ranges,

$$-R_i \leq n_i \leq R_i, \tag{10}$$

$i = 1, 2, \ldots, k$, (using the DFS algorithm [6]).

The central output frequencies of the FSBs, $\bar{f}_i$s have to be chosen appropriately to result in the desirable value of $\bar{f}_{out}$ and alow for a spectrally clean output - based on the chosen frequency mixing process. Note that $\bar{f}_{out}$ can be set with the same resolution, (7), as f_{out}.

DFS implies that with relatively small values of k and $R_1, R_2, \ldots, R_k$, the frequency step, f_{step}, can be made very small, while at the same time, the required frequency steps of the FSB, i.e. f_{in}/R_i, $i = 1, 2, \ldots, k$, can be large. The fractional resolution of the DFS scheme is R_G times larger than the geometric mean of the fractional resolutions of the constituent FSBs, where

$$R_G \ = \ \left(R_1 R_2 \cdots R_k \right)^{\frac{k-1}{k}}$$

IV. Motivating CDFS

CDFS methodology requires the same frequency ranges and resolutions from FSBs, and results in the same frequency range and resolution of the synthesizer's output signal with its predecessor DFS [6].

The advantage of CDFS is the significantly reduced (and from certain aspects - minimal) frequency ranges of the intermediate signals in the frequency mixing stages.

The following examples clarify the situation and motivate the new approach.

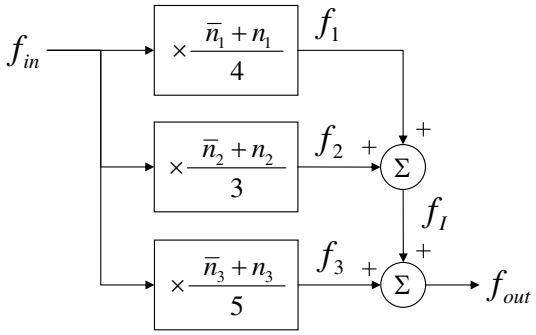

Fig. 3. A 3-FSB DFS synthesizer

A. DFS Example 1

Consider the DFS scheme in Figure 3 using three FSBs with parameters $R_1 = 4$, $R_2 = 3$ and $R_3 = 5$ (the values of $\bar{n}_i$'s are not important here as they only set $\bar{f}_i$'s, $\bar{f}_I$ and $\bar{f}_{out}$).

Using the DFS algorithm, [6], to program the values of n_i's, within their ranges (10), we generate all output frequencies (8) for $n = -R_1R_2R_3, \ldots, R_1R_2R_3$. The normalized output frequency $(f_{out} - \bar{f}_{out})/f_{in}$ ranges from -1 to $+1$, as shown in the last graph of Figure 4. The rest of the graphs show the normalized FSBs' frequencies and normalized intermediate frequency f_I.

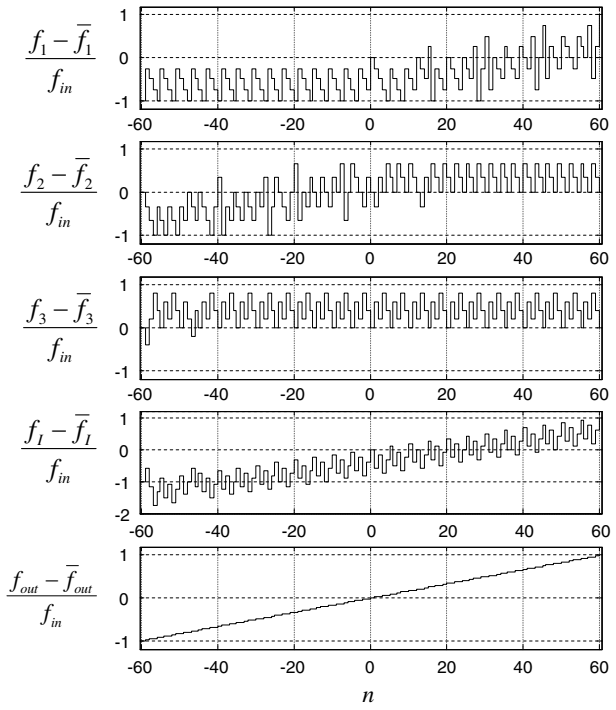

Fig. 4. Normalized FSB' and intermediate, f_I, frequencies involved in generating all output frequencies of the DFS synthesizer in Figure 3 using the DFS algorithm.

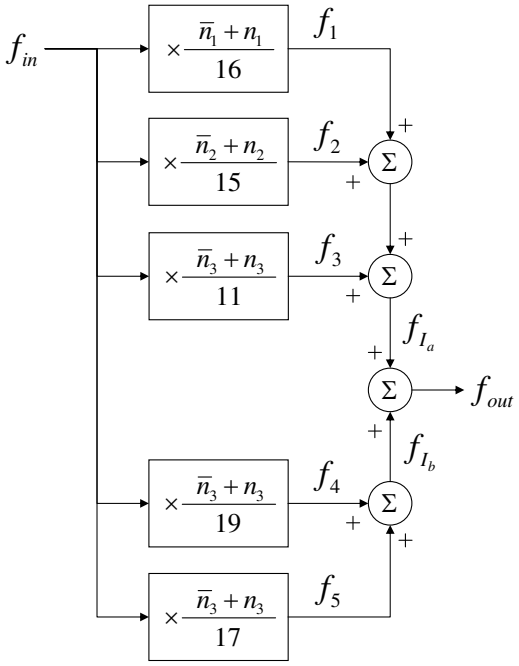

Fig. 5. A 5-FSB DFS synthesizer

Note that the normalized f_i's range within -1 and $+1$ as expected, however, the normalized intermediate frequency f_I ranges from -1.75 to $+1$ (fourth graph). Moreover, changing the values of R_1, R_2 and R_3 to $41, 56$ and 61 respectively results in the normalized intermediate frequency f_I ranging from -1.98 to $+1$.

This indicates that in general, the ranges of the intermediate frequencies f_{I_j} in the mixing stages can be significantly larger that $[\bar{f}_{I_j} - f_{in}, \bar{f}_{I_j} + f_{in}]$.

The ranges of the intermediate frequencies can increase with the number of FSBs, k, in the DFS synthesizer.

B. DFS Example 2

The ranges of the intermediate frequencies can be even larger in schemes with more FSBs, e.g. when the scheme in Figure 5 is programmed using the DFS algorithm, it results in normalized intermediate frequencies f_{I_a} and f_{I_b} ranging from -2.87 to 1 and from -0.78 to 1.89 respectively.

In contrast to DFS, the CDFS methodology and algorithm result in all normalized intermediate frequencies ranging from -1 to $+1$, i.e. the intermediate frequencies f_{I_j} in the mixing are always within $[\bar{f}_{I_j} - f_{in}, \bar{f}_{I_j} + f_{in}]$.

V. CDFS Architectures & Algorithm

The high-level general CDFS scheme is shown in Figure 6. Parameters R_i, n_i and $\bar{n}_i$ are integers as before, $\bar{n}_i$ and R_i are fixed and such that $\bar{n}_i > R_i$ and n_i is a variable taking values within the range $-R_i \leq n_i \leq R_i$, for all $i = 1, 2, \ldots, k$.

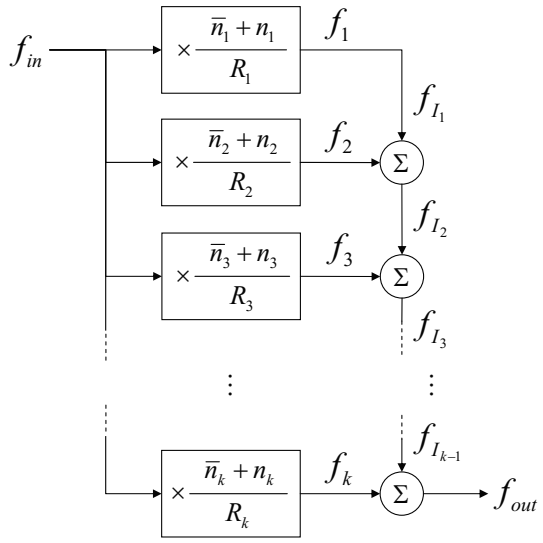

Fig. 6. Abstract high-level k-FSB CDFS scheme.

As in DFS, all FSBs are driven by the same reference signal f_{in} and their output frequencies are added or subtracted (in any chosen but fixed pattern) to produce f_{out}. Again, $\alpha_i \in \{-1, 1\}$, $i = 1, 2, \ldots, k$, are the corresponding frequency - weighting coefficients in the mixers.

The resulting output frequency, f_{out}, of the synthesizer is given by (2) and (3) and the intermediate frequencies f_{I_j} involved in the mixing process are given by[5]

$$f_{I_j} = \bar{f}_{I_j} + \left(\sum_{i=1}^{j} \alpha_i \frac{n_i}{R_i} \right) f_{in}. \tag{11}$$

where the fixed part of it, $\bar{f}_{I_j}$ is

$$\bar{f}_{I_j} = \left(\sum_{i=1}^{j} \alpha_i \frac{\bar{n}_i}{R_i} \right) f_{in}. \tag{12}$$

For convenience we define $f_{I_1} \triangleq f_1$ as well.

The foundation of CDFS is the following theorem whose proof and discussion can be found in [7].

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), and $\alpha_i \in \{-1, 1\}$, $i = 1, 2, \ldots, k$, be a fixed set of mixing weights, then:

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

$$\sum_{i=1}^{k} \alpha_i \frac{n_i}{R_i} = \frac{n}{R_1 R_2 \cdots R_k} \tag{13}$$

and inequalities (14) for all $j = 2, 3, \ldots, k$

$$\left| \sum_{i=1}^{j} \alpha_i \frac{n_i}{R_i} \right| \leq 1. \tag{14}$$

Moreover, There exists no constant $\beta < 1$ with the property that for every n with $|n| \leq R_1 R_2 \cdots R_k$ we can find a solution of (13) satisfying $|n_i| \leq \beta R_i$ for all $i = 1, 2, \ldots, k$. □

Interpreting the Theorem into frequency synthesis we have that: given the stated conditions on R_i's and α_i's we can synthesize all frequencies given by (8) and (9), by appropriately setting the values of n_i's within their ranges (10), and, at the same time satisfy

$$f_i \in \left[\ \bar{f}_i - f_{in} \ , \ \bar{f}_i + f_{in} \ \right], \tag{15}$$

for all $i = 1, 2, \ldots, k$, and

$$f_{I_j} \in \left[\ \bar{f}_{I_j} - f_{in} \ , \ \bar{f}_{I_j} + f_{in} \ \right] \tag{16}$$

for all $j = 1, 2, \ldots, k-1$.

A. CDFS Algorithm

The CDFS algorithm is summarized below (details can be found in [7]) and uses the parameterized function f defined as follows:

Let A, B be two relatively prime positive integers; given two integers, x_1 and x_2 we define $z_1 = x_1 \bmod A$, $z_2 = x_2 \bmod B$, and, $\mu = x_1 \operatorname{div} A + x_2 \operatorname{div} B$. We set

$$f_{(A,B)}(x_1, x_2) = \begin{cases} (z_1 - A, z_2 - B) & \text{if } \mu = -2 \\ (z_1 - A, z_2) & \text{if } \mu = -1 \\ (z_1, z_2) & \text{otherwise} \end{cases} \tag{17}$$

CDFS Algorithm

- STEP 0: If $n = \pm R_1 R_2 \cdots R_k$ then set $n_1 = \pm R_1$, respectively, set $n_i = 0$ for $i = 2, \ldots, k$ and STOP; otherwise proceed to STEP 1.
- STEP 1: For $i = 1, 2, \ldots, k-1$ derive, using the Euclidean algorithm, and store a solution (z_i, w_{i+1}) of the Diophantine equation

$$\frac{z_i}{R_1 R_2 \cdots R_i} + \frac{w_{i+1}}{R_{i+1}} = \frac{1}{R_1 R_2 \cdots R_{i+1}} \tag{18}$$

- 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

$$(x_{k-i}, n_{k-i+1}) = f_{(\prod_{\ell=1}^{k-i} R_\ell,\, R_{k-i+1})}\big(x_{k-i+1} z_{k-i},\ x_{k-i+1} w_{k-i+1}\big) \tag{19}$$

- STEP 3: Set $n_1 = x_1$. □

[5]Note that some intermediate frequencies may be negative (always), e.g. if $\alpha_1 = \alpha_2 = -1$ then $f_{I_2} = -f_1 - f_2$ and since it is always $f_i > 0$ for all i's, it is $f_{I_2} < 0$. In a physical implementation this simply means that the frequency sum, $f_1 + f_2$, generated by the first mixer, must be "subtracted" in the second mixer if $\alpha_2 = 1$ or added to f_3 if $\alpha_2 = -1$ and so on.

Vector $(n_1, n_2, \ldots, n_k)$ is a solution of (13), as specified in Theorem 5.1 (and satisfies (14)) when $\alpha_i = 1$ for all $i = 1, 2, \ldots, k$. For the general case one should replace $(n_1, n_2, \ldots, n_k)$ with $(\alpha_1 n_1, \alpha_2 n_2, \ldots, \alpha_k n_k)$.

B. Using the CDFS Algorithm to programm the (C)DFS synthesizer in Figure 3

The CDFS algorithm is applied to the 3-FSB scheme in Figure 3 where $k = 3$, $R_1 = 4$, $R_2 = 3$, $R_3 = 5$ and $\alpha_1 = \alpha_2 = \alpha_3 = 1$.

Based on the previous Section, the CDFS algorithm provides solutions of the Diophantine equation

$$\frac{n_1}{4} + \frac{n_2}{3} + \frac{n_3}{5} \quad = \quad \frac{n}{4 \cdot 3 \cdot 5} \tag{20}$$

that satisfy the desirable inequalities (10) and (14) - translated here as

$$-4 \le n_1 \le 4$$
$$-3 \le n_2 \le 3$$
$$-5 \le n_3 \le 5$$

and

$$\left| \frac{n_1}{4} + \frac{n_2}{3} \right| \le 1,$$

$$\left| \frac{n_1}{4} + \frac{n_2}{3} + \frac{n_3}{5} \right| \le 1$$

respectively - for all integers n such that $|n| \le 4 \cdot 3 \cdot 5$. The steps of the CDFS algorithm are:

- STEP 0: If $n = \pm 4 \cdot 3 \cdot 5 = \pm 60$ then set $n_1 = \pm 4$, respectively, set $n_i = 0$ for $i = 2, 3$ and STOP; otherwise proceed to STEP 1.
- STEP 1: Derive (some) solutions (z_1, w_2) and (z_2, w_3) of the following Diophantine equations using the Euclidean algorithm.

$$\frac{z_1}{4} + \frac{w_2}{3} \;=\; \frac{1}{4 \cdot 3}$$
$$\frac{z_2}{4 \cdot 3} + \frac{w_3}{5} \;=\; \frac{1}{4 \cdot 3 \cdot 5}$$

  Using the "gcd" function in MATLAB we get $(z_1, w_2) = (-1, 1)$ and $(z_2, w_3) = (5, -2)$.
- STEP 2: Set $x_3 = n$ and derive sequentially vectors (x_2, n_3) and (x_1, n_2) using "two parameterized versions" of function f.

$$(x_2, n_3) \;=\; f_{(4 \cdot 3,\, 5)}(x_3 z_2,\; x_3 w_3)$$
$$(x_1, n_2) \;=\; f_{(4,\, 3)}(x_2 z_1,\; x_2 w_2)$$

- STEP 3: Set $n_1 = x_1$. □

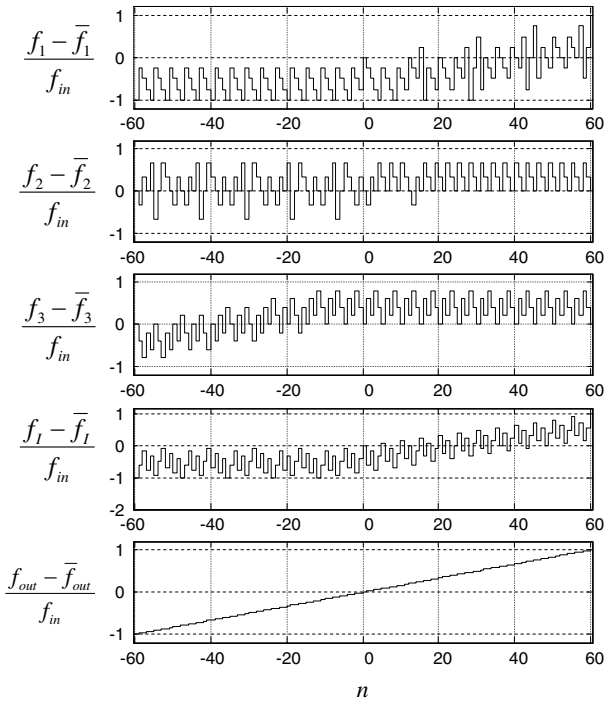

Fig. 7. Normalized FSB' and intermediate, f_I, frequencies involved in generating all output frequencies of the (C)DFS synthesizer in Figure 3 using the CDFS algorithm.

CDFS algorithm is used to program the values of n_i's, within their ranges (10) and generate all output frequencies (8) for $n = -4 \cdot 3 \cdot 5, \ldots, 4 \cdot 3 \cdot 5$.

As expected the normalized output frequency $(f_{out} - \bar{f}_{out})/f_{in}$ ranges from -1 to $+1$, as shown in the last graph of Figure 7. The rest of the graphs show the normalized FSB and intermediate f_I frequencies.

In contrast to DFS programming resulting in Figure 4, here the normalized intermediate frequency f_I (shown in the fourth graph of Figure 7) ranges from -1 to $+1$ as expected.

VI. Conclusions

The Cascaded Diophantine Frequency Synthesis (CDFS) has been presented and compared to its predecessor, Diophantine Frequency Synthesis methodology (DFS).

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

CDFS results in significantly reduced frequency ranges of the intermediate signals in the frequency mixing stages compared to (DFS) simplifying the design of the mixers and offering improved spectral purity of the output signal.

REFERENCES

[1] Daniel E. Flath "*Introduction to Number Theory*", John Willey & Sons, 1989.

[2] P. Sotiriadis, "Diophantine Frequency Synthesis A Number Theory Approach to Fine Frequency Synthesis", *IEEE International Frequency Control Symposium 2006*.

[3] William F. Egan, "*Frequency Synthesis by Phase Lock*", second edition, John Willey & Sons, 1999.

[4] Ulrich L. Rohde, "*Microwave and Wireless Synthesizers: Theory and Design*", first edition, Wiley-Interscience, 1997.

[5] Vadim Manassewitsch, "*Frequency Synthesizers*" , third edition, John Willey & Sons, 1987.

[6] Paul P. Sotiriadis, "Diophantine Frequency Synthesis, *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, Vol. 53, No. 11, Nov. 2006, pp. 1988-1998.

[7] Paul P. Sotiriadis, "Cascaded Diophantine Frequency Synthesis, *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, Vol. 55, No. 3, Apr. 2008, pp.741-751.