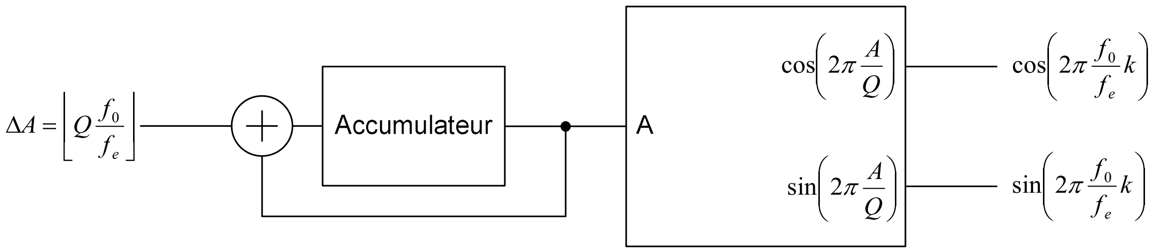
Numerically-Controlled Oscillators

1. Principle

The principle of a standard digital oscillator (or NCO for Numerically-Controlled Oscillator) of frequency f_0 with a sampling frequency f_e is described below, in the complex case :



It is based on a modulus-Q accumulator, incremented at each cycle by a value ΔA. The accumulator works in most cases in base 2 : it stores a N-bit number representing the current oscillator phase such that $Q = 2^N$ represents 2π. The phase increment can be constant (constant frequency), or follow a specific curve, which enables frequency jumps or sweeps. The current value of accumulator A is used to compute $\cos\left(2\pi\frac{A}{Q}\right)$ and $\sin\left(2\pi\frac{A}{Q}\right)$, which are the ouputs of the complex NCO. This computing can be done :

- through a pre-computed table (LUT, Look-Up Table),
- through direct calculus (use of CORDIC algorithms in FPGAs, ASICs,…)
- through combination of these methods

The NCO is essentially defined by the accumulator binary width (frequency resolution) and the output binary width (amplitude resolution).

In the optics of an implementation in digital hardware (FPGA,…), the « best » implementation method depends on accumulator binary width, output word resolution and generation rate. For low-width accumulator (say up to 10..12 bits), the LUT method should be preferred, especially if the output resolution is high.

One can also define the NCO's SFDR as the carrier / highest spur ratio (dBc). The SFDR is limited mainly by the output binary width, a bit by sine/cosine computing noise. It can be enhanced through dithering techniques which add a small white phase noise at the accumulator output in order to spread out the spurs (more spurs, but each of lesser power).

2. Extension to other bases than 2

Beside the $Q = 2^N$ case, one can imagine a NCO with $Q = b^N$, b integer. Supposing we use LUT tables, the table size can be prohibitive when N is high. A solution is to use a dual-stage computing method, by writing the accumulator value as $A = A_1 b^M + A_0$, avec $A_0 = 0 \ldots b^M - 1$ and $A_1 = 0 \ldots b^{N-M} - 1$ with $M = 1 \ldots N-1$. The two numbers A_0 and A_1 can then index two tables M_0 and M_1, defined by $M_0(k) = e^{2\pi j \frac{k}{Q}}$ with $k = 0 \ldots b^M - 1$, and $M_1(k)=e^{2\pi jk\frac{b^M}{Q}}$ with $k=0...b^{N-M}-1$. By doing the (complex) product of such indexed sub-tables outputs, we get $M_0(A_0)\cdot M_1(A_1)=e^{2\pi jA_1\frac{b^M}{Q}}e^{2\pi j\frac{A_0}{Q}}=e^{2\pi j\frac{A_1b^M+A_0}{Q}}=e^{2\pi j\frac{A}{Q}}$, which is the desired value. At the expanse of a complex multiply, the required memory has been reduced from $Q=b^N$ elements to $b^{N-M}+b^M$. In practice, the NCO accumulator is made of two registers A_0 and A_1, and the phase increment has to be precomputed modulus b^M for A_0, modulus b^{N-M} for A_1, and the carry must be handled from A_0 to A_1.

If necessary, we can iterate the decomposition until having N tables of b elements, with $N-1$ complex multiplies added.

Of course, if b is even, the tables size can be reduced by 2 through π-symmetry (sign inversion to manage). Same if b is a multiple of 4, through $\frac{\pi}{2}$ rotation (sign inversion + real-imaginary swap to manage).

The modulus b ($b\neq 2$) arithmetic may be not as easy in the general case and can limit the NCO's data rate.

3. Extension to the product of two relatively prime integers

If Q is not prime or a power of a prime integer, we can write $Q=UV$, with U et V relatively prime. A modulus-Q NCO can then be implemented with two NCOs working modulus-U and modulus-V. These elementary NCOs allow the generation of frequencies $f_u=u\frac{f_e}{U}$, $u=0...U-1$, and $f_v=v\frac{f_e}{V}$, $v=0...V-1$.

The (complex) product of these NCOs outputs allows the generation of the frequencies $f_{u,v}=u\frac{f_e}{U}+v\frac{f_e}{V}$, which can rewritten as $f_{u,v}=f_e\left(\frac{u}{U}+\frac{v}{V}\right)=(u\cdot V+v\cdot U)\frac{f_e}{UV}=(u\cdot V+v\cdot U)\frac{f_e}{Q}$.

Since U and V are relatively prime, there are p and q such that $p\cdot V+q\cdot U=1$ (Bézout's theorem).

p and q can be obtained for example through the extended Euclid's algorithm $(p,q,r)=PGCD_e(U,V)$, which is :

$$\begin{cases} r_0=V \\ p_0=1 \\ q_0=0 \\ r_0'=U \\ p_0'=0 \\ q_0'=1 \end{cases} \text{ and as long as } r_i'\neq 0 : \begin{cases} s_{i+1}=r_i\div r_i' \\ r_{i+1}=r_i' \\ p_{i+1}=p_i' \\ q_{i+1}=q_i' \\ r_{i+1}'=r_i-s_{i+1}r_i'=r_i \bmod r_i' \\ p_{i+1}'=p_i-s_{i+1}p_i' \\ q_{i+1}'=q_i-s_{i+1}q_i' \end{cases}$$

At end of execution, we can take : $p=p_i$, $q=q_i$ and $r=r_i$. We have then : $pV+qU=r=PGCD(U,V)$, ie $pV+qU=r=1$ if U and V relatively prime.

In practice, one can take $p=U+p_i$ (resp. $q=V+q_i$) if p_i (resp. q_i) is negative, to avoid handling negative number in the NCOs.

The frequency $f_c=f_e\frac{P}{Q}$ with $P=0...Q-1$ can then be generated by choosing :

$u=(P\cdot p)\bmod U$ and $v=(P\cdot q)\bmod V$

4. General case

In the general case, the implementation of a NCO generating the exact frequencies $\frac{P}{Q} f_e$, P and Q positive integers with $P < Q$, can be done with the procedure below :

- decomposition of Q in prime factors : $Q = \prod_{m=1}^{M} Q_m^{R(m)}$, where the Q_m are M distinct prime integers.
- implementation of a bunch of M elementary NCOs working modulus $Q_m^{R(m)}$.
- Computation of the phase increments u_m for each of the elementary NCOs, through $M-1$ applications of the extended Euclid's algorithm, with for example :

$$v_0 = P \text{ et } \begin{cases} U_m = Q_m^{R(m)} \\ V_m = \prod_{k=m+1}^{M} Q_k^{R(k)} \\ P_m = v_{m-1} \\ (p_m, q_m, r_m) = PGCD_e(U_m, V_m) \\ u_m = (P_m p_m) \bmod U_m \\ v_m = (P_m q_m) \bmod V_m \end{cases} \quad \text{with } m = 1..M-1 \text{, then } u_M = v_{M-1}.$$

This implementation principle is limited by the maximum size of the elementary tables, which is $\max_m (Q_m)$, ie the greatest prime factor of Q. Moreover, if an elementary NCO is itself implemented as a group of NCOs of same prime factor, an additional step is needed to compute the phase increment of each of these « sub-NCO ».

The computing of a single output sample from the global NCO can take a non-negligeable number of operations. In case of a FPGA implementation, these operations can be pipelined, and the output data rate can be kept high.

5. Extension to the product of M numbers

By enabling the propagation of carries between the accumulators of several NCOs, and using tables of fractions of turn, one can build a global NCO of size equal to the product of the sizes of the basic NCOs. The carries propagation constraint can lead to a reduction of the working frequency in an FPGA.

The generation of a signal at frequency $\frac{P}{Q} f_e$, with $Q = \prod_{m=1}^{M} Q_m$, P et Q positive integers, and $P < Q$, can be done through the following procedure :

Let NCO of index 0 the least significant one, the NCO of index $M-1$ is the most significant.
The values tables of the basic NCOs are defined by :

$$T_m(k) = e^{j2\pi \frac{k}{\prod_{l=m}^{M-1} Q_l}} \text{ with } 0 \le k < Q_m.$$

Except for the most significant NCO, the NCO m table contains Q_m values for the angles 0 to the minimal phase increment of NCO $m+1$, ie $\frac{2\pi}{\prod_{l=m+1}^{M-1} Q_l}$.

Let $a_m(n)$ the current value of accumulator of the index m NCO; we have : $0 \le a_m(n) < Q_m$.

Let $d_m(n)$ the phase increment of the index m NCO; we have : $0 \le d_m(n) < Q_m$.
For sample n, the output of the global NCO is the product of the basic NCOs outputs :

$$y(n) = \prod_{m=0}^{M-1} T_m(a_m(n)) = \prod_{m=0}^{M-1} e^{j2\pi \frac{a_m(n)}{\prod_{l=m}^{M-1} Q_l}} = e^{j2\pi \sum_{m=0}^{M-1} \frac{a_m(n)}{\prod_{l=m}^{M-1} Q_l}} = e^{j2\pi \frac{a_0(n) + \sum_{m=1}^{M-1}\left[a_m(n)\prod_{l=0}^{m-1} Q_l\right]}{\prod_{l=0}^{M-1} Q_l}} = e^{j2\pi \frac{a_0(n) + \sum_{m=1}^{M-1}\left[a_m(n)\prod_{l=0}^{m-1} Q_l\right]}{Q}}$$

For sample n, the phase increment of the global NCO is :

$$d(n) = d_0(n) + \sum_{m=1}^{M-1}\left[d_m(n)\prod_{l=0}^{m-1} Q_l\right], \text{ with } 0 \le d(n) < Q.$$

The accumulators values for sample n+1 are : $a_m(n+1) = a'_m(n+1) \bmod Q_m$, with :

$$a'_0(n+1) = a_0(n) + d_0(n)$$
$$a'_1(n+1) = a_1(n) + d_1(n) + \delta\left(a'_0(n+1) \ge Q_0\right)$$
…
$$a'_{M-1}(n+1) = a_{M-1}(n) + d_{M-1}(n) + \delta\left(a'_{M-2}(n+1) \ge Q_{M-2}\right)$$

with $\delta(v)$ =1 if v is true, 0 otherwise (carry).

6. Case of the frequency ramp

We want here to generate a frequency ramp (the frequency varying linearly with time) through a NCO with a modulus-Q accumulator. This is done by modifying the phase increment at each sample.

The value of NCO's accumator A and its increment are :

$$\begin{cases} A(n) = \left[A(n-1) + \Delta A(n)\right] \bmod Q \\ \Delta A(n) = \Delta A(n-1) + \Delta^2 A \end{cases}$$

$\Delta A(n)$ is the current phase increment

$\Delta^2 A$ is the frequency increment (supposed constant for a linear frequency ramp)

One can write :

$$A(n) = \left[A(n-1) + \Delta A(0) + n\Delta^2 A\right] \bmod Q$$

$\Delta A(0)$ is the initial phase increment (starting frequency)

or :

$$A(n) = \left[A(0) + n\Delta A(0) + \frac{n(n-1)}{2}\Delta^2 A\right] \bmod Q$$

$A(0)$ is the initial phase

$\Delta A(0)$ is the initial phase increment (starting frequency)

For a N-samples ramp, the frequency amplitude is $(N-1)\Delta^2 A$ (difference between endind and starting frequencies).

With a sampling frequency f_e, an initial phase φ_0, a starting frequency f_0, an ending frequency f_1, a ramp duration T_r, we have :

$$A(0) = round\left(Q\frac{\varphi_0}{2\pi}\right)$$

$$\Delta A(0) = round\left(Q\frac{f_0}{f_e}\right)$$

$$N = round\left(f_e T_r\right)$$

$$\Delta^2 A = round\left(Q\frac{f_1 - f_0}{(N-1)f_e}\right)$$

Adding the constraint of phase continuity between the start and end of the ramp (to get a nicer signal in case of a periodic ramp, where the initial values are reloaded each N samples), we have : $A(N) = A(0)$,

ie $A(0) = \left[A(0) + N\Delta A(0) + \frac{N(N-1)}{2}\Delta^2 A\right] \bmod Q$

or $\left[N\Delta A(0) + \frac{N(N-1)}{2}\Delta^2 A\right] \bmod Q = 0$

A particular sufficient condition is :

Q is multiple of $N(N-1)$

and $\Delta A(0)$ is multiple of $\frac{Q}{N}$

and $\Delta^2 A$ is multiple of $2\frac{Q}{N(N-1)}$

which can also be written as :

$Q = KN(N-1)$

and f_0 multiple of $\frac{f_e}{N}$

and (f_1- f_0) multiple of $2\frac{f_e}{N}$

Legal stuff :