Simulation of a parametric quartz crystal oscillator by the symbolic harmonic method

Nicolas Ratier, Michaël Bruniaux, Serge Galliou, Rémi Brendel

Simulation of a parametric quartz crystal oscillator by the symbolic harmonic method

N. Ratier,* M. Bruniaux, S. Galliou, and R. Brendel
Institut FEMTO–ST, département LPMO & LCEP, CNRS UMR 6174
32 avenue de l'Observatoire, F-25044 Besançon, France

The Symbolic Harmonic Analysis (SHA) method is a frequency domain approach for computing the steady state of ultra stable quartz crystal oscillators. First, we present a general method to convert a set of differential equations into a system of non-linear algebraic equations that can be solved for the periodic steady state solution. Then, we apply this method to simulate the behavior of a parametric quartz crystal oscillator currently being developed at FEMTO–ST/LCEP. This oscillator uses a 10 MHz quartz resonator and a 20 MHz signal derived from the oscillator nonlinearities is used to pump a varactor-diode. The quartz resonator time constant induces too long simulation time with classical integration methods. So that the symbolic harmonic analysis offers a good alternative to quickly obtain the periodic steady state. Simulation results are compared with experimental data.

I. INTRODUCTION

The work described in this paper presents the latest development of a simulation method to determine in real time the steady–state solution of ultra–stable crystal oscillators.

Usually the method used to simulate this kind of circuit is the so–called "harmonic–balance" method [1]. Roughly, this numerical method amounts to compute the behavior of the linear part of the circuit in the frequency domain and the nonlinear part in the time domain. The name stems from an approach based on current balancing between the linear and nonlinear parts.

In our method, by using symbolic calculation, the system of nonlinear differential equations describing the oscillator circuit is replaced by a system of nonlinear equations of Fourier coefficients whose solution is an approximation of the steady–state response of the circuit. The harmonic analysis method imposes the steady–state conditions through a Fourier expansion of the unknown functions. This method in which the simulation time no longer depends on the transient is used to develop a real time simulation tool for ultra–stable oscillator circuits.

The main drawback of symbolic methods lies in the fact that they usually involve a very large number of terms. The first part of the paper proposes a solution based on "tree parsing" to solve this problem, while the second part applies the method to simulate a parametric oscillator.

*Electronic address: nicolas.ratier@lpmo.edu

II. PRINCIPLE OF THE SYMBOLIC HARMONIC ANALYSIS METHOD

$P(T_0)$ denotes the set of all periodic functions of bounded variation with period T_0. The system of differential equations under consideration is of the form (Eq. 1) where $u \in P(T_0)$ is the stimulus waveform, x is the unknowns waveform to be found and f is continuous and real.

$$f(x, x', u) = 0 \tag{1}$$

If the solution x exists, is real, and belongs to $P(T_0)$, it can be written as a Fourier series (Eq. 2) where $\omega_0 = 2\pi/T_0$.

$$x(t) = X_0 + \sum_{k=1}^{\infty} X_k \cos(k\omega_0 t) + \sum_{k=1}^{\infty} X_{-k} \sin(k\omega_0 t) \tag{2}$$

Since $x \in P(T_0)$ and $u \in P(T_0)$ imply $x' \in P(T_0)$ and $f(x, x', u) \in P(T_0)$, by substituting x, its derivative and u into f, the resulting equation can be written under a Fourier series form (Eq. 3).

$$f(x, x', u) = F_0 + \sum_{k=1}^{\infty} F_k \cos(k\omega_0 t) + \sum_{k=1}^{\infty} F_{-k} \sin(k\omega_0 t) \tag{3}$$

By using the orthogonality of sinusoidal functions, (Eq. 3) can be rewritten as a system of nonlinear equations (Eq. 4), one for each harmonic defined by the assumed solution. Then the nonlinear system (4) can be solved numerically to obtain the Fourier coefficients X_k of the unknowns.

$$F_k(X_k) = 0 \text{ for all } k \in Z \tag{4}$$

Nevertheless, even by truncating the Fourier series, a direct application by symbolic calculation of the method described above to build the nonlinear system leads to an exponential growth of the number of terms.

The method proposed to overcome this difficulty is to rewrite each differential equation of the system as a binary tree so that each node involves only one algebraic operation. Then the trees are progressively reduced in harmonic (i.e. Fourier series) form from the bottom to the top: at each node new coefficients expressed in function of the previous ones are generated. Eventually, it remains only a Fourier series with manageable coefficients.

The mapping of differential equations into binary trees is defined by the following abstract syntax: the `UNKNOWN` terminals are elements of the unknown vector $x(t)$ and the `FUNCTION` terminals are either constants or functions already in harmonic form, as the stimulus waveform $u(t)$ does.

```
type odeTree =
       SUM   of odeTree * odeTree
     | PROD  of odeTree * odeTree
     | POWER of odeTree * INTEGER
     | DIFF  of odeTree
     | EXP   of odeTree
     | COS   of odeTree
     | SIN   of odeTree
     | UNK   of UNKNOWN
     | FCT   of FUNCTION
```

Because only `EXP`, `COS` and `SIN` functions of Fourier series can be expanded into Fourier series, the grammar accepts only these transcendental functions. Any other transcendental function (log, tan, ...), inverse of a function, and composite function are rejected. To fit all ODE to the given grammar, one proceeds by introducing an additional unknown to the equation system, which transforms the initial differential equation system into an algebro–differential system. This handling is trivial as shown in the following example.

$$\cdots + \tan V_1 + \cdots + \arccos V_2 + \cdots = 0 \qquad (5)$$

becomes

$$\cdots + H_1 + \cdots + H_2 + \cdots = 0 \qquad (6)$$

$$H_1 \cos V_1 = \sin V_1 \qquad (7)$$

$$\cos H_2 = V_2 \qquad (8)$$

The latter equations are now recognized by the grammar. The same method can be directly applied to the inverse of a function and to the composite functions.

III. EVALUATION OF `UNK`, `FCT`, `SUM`, `PROD`, `DIFF`, `EXP`, `COS`, `SIN` NODES

Once the tree of the differential equation is built, each node of the tree can be progressively reduced (or evaluated). Assuming that $V_1(t)$ is an unknown function of the differential system and R_1 is a parameter, the reduction is performed as follow:

- `UNK(V1(t))` is replaced by its Fourier series limited to N terms, i.e. it returns:

```
  A00V1
+ A01V1 cos  wt + B01V1 sin  wt
+ A02V1 cos 2wt + B02V1 sin 2wt
+ A03V1 cos 3wt + B03V1 sin 3wt
```

- `FCT(R1)` is already under the right form, so it is replaced by itself.

- `SUM(S1,S2)` with

```
S1 = A00S1 + A01S1 cos wt
           + B01S1 sin wt + ...
S2 = A00S2 + A01S2 cos wt
           + B01S2 sin wt + ...
```

  is reduced by generating new coefficients

```
TT1 = A00S1 + A00S2
TT2 = A01S1 + A01S2
TT3 = B01S1 + B01S2
... = ...
```

  and returns:

```
TT1 + TT2 cos wt + TT3 sin wt + ...
```

- `PROD(S1,S2)`, `DIFF(S1)` are reduced in the same way as `SUM(S1,S2)`, but with a bit more complex generated coefficients.

- The "harmonization" of `EXP(S1)`, `COS(S1)` and `SIN(S1)` functions is explained in [2]. The solution of this problem is quite similar to the harmonization of ODE. A binary tree is constructed and new coefficients are generated as a function of the previous ones during the tree parsing. Each transcendental function has a different associated tree.

IV. EVALUATION OF POWERS (`POWER`)

Given x and n, where x is a Fourier series and n is a positive integer, we study the problem of computing symbolically `POWER(x,n)`, or x^n, efficiently. Here, "multiplication" means multiplication of series. Although we are concerned with multiplication of powers of x, the problem can be reduced to addition, since the exponents are additive. This leads to the following abstract formulation:

An addition chain for the integer number n is a sequence of integers

$$a_0 = 1, a_1, a_2, \ldots, a_r = n \qquad (9)$$

with the property that, for all $i = 1, 2, \ldots, r$:

$$a_i = a_j + a_k, \text{for some} k \leqslant j < i \qquad (10)$$

This means that each exponentiation in the chain can be evaluated by multiplying two of the previous exponentiation results. The optimal way to compute x^n by multiplication is given by the addition chain for n having the smallest length r. The smallest length r for which an addition chain for n exists is denoted by $l(n)$.

Despite numerous work in this area, the determination of a minimal–length addition chain generating the desired exponent is still an open problem is mathematics. The weaker problem to compute $l(n)$ is neither solved. However, a lower and a upper bounds of $l(n)$ are known.

$$\lceil \log_2 n \rceil \leqslant l(n) \leqslant \lfloor \log_2 n \rfloor + \nu(n) - 1 \qquad (11)$$

where $\lceil \log_2 n \rceil$ is the ceiling of x (smallest integer greater than or equal to x), $\lfloor \log_2 n \rfloor$ is the floor of x (greatest integer less than or equal to x) and $\nu(n)$ is the number of 1's in the binary representation of n.

Our SHA program use the following recursive algorithm to compute x^n for a Fourier series x and a positive integer n. This algorithm is known as the square–and–multiply algorithm [3]. The repeated application of this algorithm amounts to decompose the exponent into a sequence of squares and products. It requires only one temporary storage x (and of course the current partial result).

$$\text{Power}(x,n) = \begin{cases} x, & \text{if } n = 1 \\ \text{Power}(x^2, n/2), & \text{if } n \text{ is even} \\ x \times \text{Power}(x^2, (n-1)/2), & \text{if } n \text{ is odd} \end{cases} \qquad (12)$$

The square–and–multiply algorithm is used for two main reasons:

1. Although, it doesn't lead to an optimal addition chain, it is quite efficient. Compared to the ordinary method of multiplying x with itself $n-1$ times, this algorithm uses only $O(\log_2 n)$ multiplications. (The number of multiplications required to compute $x^n$by the square–and–multiply algorithm is exactly $\lfloor \log_2 n \rfloor + \nu(n) - 1$).

2. Unlike optimal addition chains, the addition chains computed by the square–and–multiply algorithm have always a binary tree structure. So the computation of x^n by this algorithm is naturally integrated into the SHA method as it is based on binary trees representation.

For example, table I shows the application of the square–and–multiply algorithm on x^9 and x^{10}. The second column follows the application of the algorithm giving the addition chains in reverse order, and the third column gives the algebraic equivalent.

The computation of x^9 and x^{10} are represented by the following binary trees (Fig. 1). It should be noted that each power of x has a different binary tree and two closed powers, for example x^n and x^{n+1}, don't have closed binary trees at all. The trees are reduced as explains in the previous section.

Power	Algorithm	Algebraic
x^9	$9, 8, 4, 2, 1$	$((x^2)^2)^2 x$
x^{10}	$10, 5, 4, 2, 1$	$((x^2)^2 x)^2$

TABLE I: Computation of x^9 and x^{10}

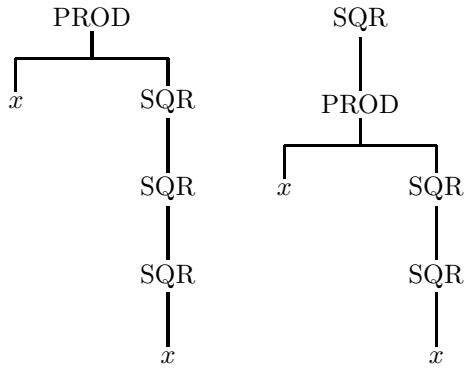

FIG. 1: Binary tree of x^9 and x^{10}

Table II illustrates the efficiency of the square–and–multiply algorithm to compute x^n. $l(n)$ is the minimal number of multiplications. For large n, the lower and upper bounds and computed by inequality (Eq. 11). $l^\star(n)$ is the number of multiplications required by the square–and–multiply algorithm. Results have to be compared to the ordinary method of powering which needs $n-1$ multiplications.

n	$l(n)$	$l^\star(n)$
10	4	4
100	8	8
1000	$10 \leqslant l(n) \leqslant 14$	14
10000	$14 \leqslant l(n) \leqslant 17$	17
100000	$17 \leqslant l(n) \leqslant 21$	21
1000000	$20 \leqslant l(n) \leqslant 25$	25
10000000	$24 \leqslant l(n) \leqslant 30$	30
100000000	$27 \leqslant l(n) \leqslant 37$	37
1000000000	$30 \leqslant l(n) \leqslant 41$	41

TABLE II: Efficiency of the square–and–multiply algorithm

V. APPLICATION: PARAMETRIC OSCILLATOR

The symbolic harmonic method is applied to analyze the forced–mode behavior of the parametric quartz oscillator (PXO) shown in Fig. 2. The principle of this oscillator developed at FEMTO–ST/LCEP is described in [4]. It uses a 10MHz quartz and a 20MHz pump U_{pmp} coming from the second harmonic at 20 MHz generated by a varactor-diode. For instance the parameters of the oscillator are $R_1 = 100\Omega$,

$$U_b + U_{\rm vd}(t) + R_2\frac{d}{dt}(C_{\rm vd}(t).U_{\rm vd}(t)) + L.\frac{d^2}{dt^2}(C_{\rm vd}(t).U_{\rm vd}(t)) = \quad (13)$$
$$-R_1\frac{d}{dt}(C_{\rm vd}(t).U_{\rm vd}(t)) - R_1C_q\frac{d}{dt}U_q(t) + V_p\sin(2\omega_q t)$$
$$U_q(t) + R_qC_q\frac{d}{dt}U_q(t) + L_qC_q\frac{d^2}{dt^2}U_q(t) = \quad (14)$$
$$-R_1\frac{d}{dt}(C_{\rm vd}(t).U_{\rm vd}(t)) - R_1C_q\frac{d}{dt}U_q(t) + V_p\sin(2\omega_q t)$$
$$C_{\rm vd}(t).H(t) = C_{0\,\rm vd} \quad (15)$$
$$1 + \frac{U_{\rm vd}(t)}{\phi_0} = H^2(t) \quad (16)$$

$R_2 = 300\Omega$, $U_b = -1V$, $U_{\rm pmp} = 4.5V$, and L is tuned at 10MHz with the varactor–diode capacitance value. The resonator is a 10MHz SC cut quartz crystal, exhibiting a unloaded quality factor of about 1 10^6.

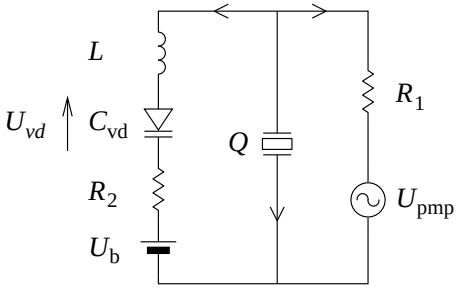

FIG. 2: Quartz crystal Parametric oscillator

The voltage–capacitance relationship of a varactor–diode is written [5] as Eq. (17), where $C_{\rm vd\,0}$ is the diode capacitance at $U_{\rm vd} = 0$ (typically a few tens of picofarad), ϕ_0 is the contact potential, and γ is the fractional change in capacitance ($\gamma = -1/2$ for abrupt junction, and $\gamma = -1/3$ for graded–junction varactors).

$$C_{\rm vd} = C_{0\,\rm vd}\left(1 + \frac{U_{\rm vd}}{\phi_0}\right)^{\gamma} \quad (17)$$

The circuit is described by the system of ordinary differential equations (Eq. 13) to (Eq. 16). The "harmonization" of Eq. 17 is done by introducing an additional unknown function $H(t)$. The equation of the varactor diode, for $\gamma = -1/2$ is then replaced by the two Eq. 15 and 16.

The unknowns of the system are the voltages across the varactor $U_{\rm cd}(t)$ and the quartz series capacitance $U_q(t)$, the capacitance $C_{\rm vd}(t)$ and the additional function $H(t)$. $H(t)$ doesn't have any physical meaning. The previous ODE system is then solved by SHA method by replacing all unknowns by Fourier series, building and reducing ODE trees. The resulting equations involving Fourier coefficients of the unknowns are then solved numerically. The varactor voltage $U_{\rm vd}$ versus time in plotted in Fig. 3.

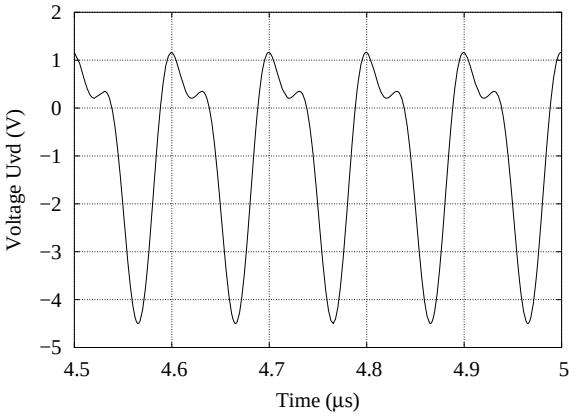

FIG. 3: SHA Simulation of the varactor–diode voltage $U_{\rm vd}$ versus time.

Fig. 4 shows experimental results performed in forced mode [6]. Theoretical results are in good agreement with experimental ones, although theoretical improvements still have to be done. The simulated curve plotted in Fig. 3 is computed in a few seconds. Numerical computation of the system (Eq. 13 to Eq. 16) done by usual techniques [6] need a few hours and give results not as good as the SHA method.

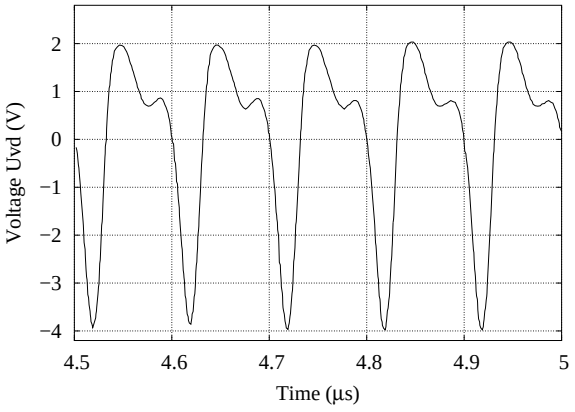

FIG. 4: Experimental varactor–diode voltage $U_{\rm vd}$ versus time.

VI. CONCLUSIONS

The principle of a symbolic–numeric method to analyze ultra–stable oscillators, presented in this paper and [2], allows one to replace a system of nonlinear differential equations by a system of nonlinear equations of Fourier coefficients, whose solution is an approximation of the steady–state solution. The method has been successfully applied to simulate a parametric oscillator.

A solution based on trees to manage the large number of coefficients inherent to symbolic computation has been proposed. At the opposite of all other harmonic methods, the linear and the nonlinear parts of the differential equation are processed in an uniform way in the Fourier domain.

Using symbolic computation technique is not common in the electronics circuit community, and in the time–frequency domain, whereas it proved its ability to solve quickly and efficiently the difficult problem of the simulation of quartz crystal oscillators. The gain in terms of computing time is in the order of decades for usual oscillator circuits.

The calculation times to solved the equations generated in the last step of the method is independent on the length of transients, because the symbolic harmonic method imposes the steady–state conditions by virtue of Fourier expansion of the unknowns.

No mention has been made about how to solve the system of nonlinear equations generated in the last step of the symbolic harmonic method. The resulting system is highly nonlinear and sparse. Efforts are currently made to develop specific and efficient numerical algorithms in this direction.

[1] K. S. Kundert, J. K. White, and A. Sangiovanni-Vincentelli, *Steady–State Methods for Simulating Analog and Microwave Circuits* (Kluwer Academic Publishers, 2003).

[2] N. Ratier, M. Bruniaux, S. Galliou, and R. Brendel, in *Proc. of the 19th European Frequency and Time Forum (EFTF)* (Besançon, France, March 2005), pp. 413–418.

[3] D. E. Knuth, *The art of computer programming, Vol II. Seminumerical algorithms* (Addison–Wesley, 1998), ISBN 0-201-89684-2.

[4] V. Komine, S. Galliou, and A. Makarov, IEEE Trans. Ultrason., Ferroelect., Freq. Contr. **50**, 1656 (2003).

[5] L. Blackwell and K. Kotzebue, *Semiconductor Diode Parametric Amplifiers* (Englewood Cliffs, N.J, Prentice–Hall, 1961).

[6] S. Galliou, R. Brendel, N. Ratier, M. Bruniaux, P. Abbe, and G. Cibiel, in *Proc. of the 19th European Frequency and Time Forum (EFTF)* (Besançon, France, March 2005), pp. 408–412.