View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by HAL-CEA

Synchronization Analysis of Networks of Self-sampled All-Digital Phase-Locked Loops

Jean-Michel Akre, Jérôme Juillard, Dimitri Galayko, Eric Colinet

To cite this version:

Jean-Michel Akre, Jérôme Juillard, Dimitri Galayko, Eric Colinet. Synchronization Analysis of Networks of Self-sampled All-Digital Phase-Locked Loops. IEEE Transactions on Circuits and Systems Part 1 Fundamental Theory and Applications, Institute of Electrical and Electronics Engineers (IEEE), 2012, 59 (4), pp.708-720. <10.1109/TCSI.2011.2169745>. <hal-00695843>

HAL Id: hal-00695843

https://hal-supelec.archives-ouvertes.fr/hal-00695843

Submitted on 5 Oct 2012

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Synchronization Analysis of Networks of Self-Sampled All-Digital Phase-Locked Loops

J. M. Akré, J. Juillard, D. Galayko, and E. Colinet, *Member, IEEE*

***Abstract*— This paper analyses the stability of the synchronized state in Cartesian networks of identical all-digital phase-locked loops (ADPLLs) for clock distribution applications. Such networks consist in Cartesian grids of digitally-controlled oscillator nodes, where each node communicates only with its nearest neighbors. Under certain conditions, we show that the whole network may synchronize both in phase and frequency. A key aspect of this study lies in the fact that, in the absence of an absolute reference clock, the loop-filter in each ADPLL is operated on the irregular rising edges of the local oscillator and consequently, does not use the same operands depending on whether the local clock is leading or lagging. Under simple assumptions, these networks of so-called "self-sampled" all-digital phase-locked-loops (SS-ADPLLs) can be described as piecewise-linear systems, the stability of which is notoriously difficult to establish, The main contribution of this paper is a simple design rule that must be met by the coefficients of each loop-filter in order to achieve synchronization in a Cartesian network of arbitrary size. Transient simulations indicate that this necessary synchronization condition may also be sufficient for a specific (but important) class of SS-ADPLLs. A synthesis of the different approaches that have been conducted in the study of the synchronization of SS-ADPLLs is also done.**

I. Introduction

In large-scale synchronous systems-on-chips (SOCs), clock distribution systems of synchronized oscillators [1-2] are an alternative approach to classical tree-like clock distribution methods [3-4]. In such systems, a network of synchronized oscillators deals with the distribution of time and frequency over a wide geographical area. The goal of the designer of the network is then to guarantee that the time and frequency scales of all the clocks are aligned after a finite time. The subject of synchronization - and, more generally, the subject of consensus - has received considerable interest in past years: good entry points are [5], mostly dedicated to natural synchronization phenomena, and [6-7], more specifically devoted to network synchronization.

The system considered here is composed of N nodes of identical all-digital phase-locked loops (ADPLLs), each of which may be regarded as an oscillator trying to match the phase of its neighbors. With such an oscillator at the core of each synchronous area of a SOC, the synchronization between all neighboring synchronous areas can be guaranteed, and thus the synchronization of the entire system. This approach solves some of the problems inherent to the traditional approaches (e.g. H-tree), which suffer among others from poor scalability and high skew. The reliability of this approach was established in 1995 by Pratt and Nguyen [1]. An implementation was proposed by Gutnik and Chandrakasan [2] in 2000. Nevertheless, the PLL implemented by them suffered from the drawbacks associated with the use of analog techniques. The HODISS project, funded by the ANR ARFU program, aims at pursuing the work in [1] and [2] by performing a clock distribution system based on a fully digital design flow, in order to be easily integrated, compatible with the functional digital blocks of the chips and to benefit from the noise-immunity of digital components.

The present work is the continuation of some previous papers in which so-called "self-sampled" ADPLLs (SS-ADPLLs) were introduced [8-9]. A typical SS-ADPLL can be broken down into three components: a digital phase detector (DPD), a digital filter and a digitally-controlled oscillator (DCO) (Fig. 1). The DPD outputs a signal which is proportional to the time difference between the rising edges of the local clock and of the neighbor clock. Its main characteristic is that, in the absence of an absolute reference clock, the loop filter is operated on the rising edges of the local time-varying clock. Consequently, the loop filter does not use the same operands depending on whether the local clock is leading or lagging: for example, when the local clock leads, the output of the DPD is updated after the loop filter is operated (Fig. 2), because the rising edge of the neighbor clock has not been received yet. In this respect, our work differs from many recent works on the subject of ADPLLs, such as [10-12], which are focused on the nonlinearity of the DPD and neglect the influence of the sequencing of events, as opposed to [8-9,13].

In the current study, which synthesizes and extends our previous works to Cartesian networks of arbitrary size, we investigate what filter coefficients to choose in order to ensure stability and, hence, synchronization. It has been shown in [8-9], that the study of the synchronization of one SS-ADPLL, or of a network of such devices, breaks down to the study of the

Manuscript received March 29, 2011. This work was supported by the French National Agency of Research (ANR) through the HODISS project.

J. M. Akré and J. Juillard are with E3S, SUPELEC, Plateau de Moulon, 3 rue Joliot-Curie, 91190 Gif-sur-Yvette, France (e-mail: jerome.juillard@supelec.fr, jean-michel.akre@supelec.fr).

D. Galayko is with LIP6, UPMC, Université Paris 6, 4, place Jussieu 75252 Paris cedex 05, France (e-mail: dimitri.galayko@lip6.fr).

E. Colinet is with CEA-LETI, MINATEC, 17 rue des martyrs, 38054 Grenoble cedex 9, France (e-mail: eric.colinet@cea.fr).

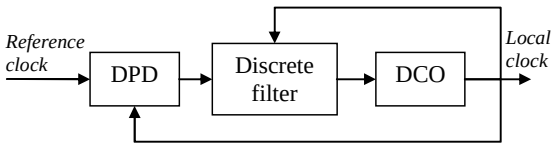


Fig. 1 - Block-diagram of a self-sampled PLL.

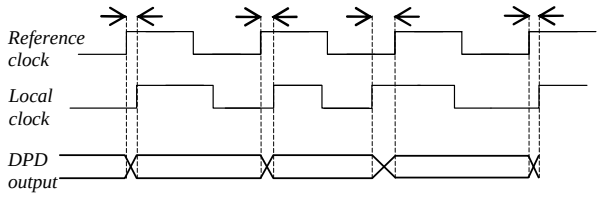


Fig. 2. - When the local clock leads, the output of the DPD is updated on a rising edge of the reference clock, after the filter is operated.

stability of a piecewise-linear system (PLS), parameterized by the coefficients of the loop filter and the number of nodes in each line and column of the Cartesian network. There usually exists no analytical means of deriving necessary and sufficient conditions for the stability of such a family of PLS. On the other hand, given an individual of this family, one may test for its stability in different ways. In our previous work, we have explored two of these methods, neither of which proves completely satisfactory. The first approach is through transient simulation of the PLS, which can easily be performed for a network of arbitrary size. However, it is only possible to establish stability in this manner for a given finite set of initial conditions. The second approach is based on the construction of a piecewise-quadratic Lyapunov function (PQLF) [14-15]. For a given PLS, one may solve a system of linear matrix inequalities (LMIs) to determine whether a PQLF exists, in which case the system is stable. However, there are several problems with this approach: first of all, the existence of a PQLF is only a sufficient condition for the stability of a PLS. For example, there are several cases of stable SS-ADPLLs for which no PQLF can be constructed, as shown in [9]. Moreover, solving the system of LMIs becomes computationally intensive as the size of the network increases.

The main contribution of this paper is the derivation of a necessary condition for the stability of a Cartesian network of SS-ADPLLs. In the case when the loop filter is a discrete PI filter, parameterized by two coefficients, we show how to determine the region of the parameter-space in which the coefficients must be chosen as a prerequisite for stability, regardless of network size. This results in simple design rules, which can easily be extended to more general filters. This necessary condition may or may not be sufficient, depending on the particular type of SS-ADPLL being used.

In section II, the general model of a single SS-ADPLL and its governing equations are presented. In particular, we show how differing implementations of the loop filter result in different system dynamics. Section III is dedicated to Cartesian networks of SS-ADPLLs. It is shown that the filter coefficients must satisfy a certain condition in order for the network to be stable. Interestingly, this condition is independent of network size. In section IV, the validity of our theoretical results is illustrated with simulation results.

II. General model of SS-ADPLLs

An SS-ADPLL is represented in Fig. 1. It is composed of a digital phase detector (DPD), a digital loop filter and a digitally-controlled oscillator (DCO). Throughout this paper, it is assumed that the loop filter is a proportional-integral (PI) filter. The use of a PI filter in a classical PLL design makes it possible to synchronize both in frequency and in phase [16]. The PI filter is driven by the rising edges of the output of the DCO (the local clock). The DPD is assumed to be linear and outputs a code proportional to the value of the corresponding timing error, i.e. to the time elapsed between a rising edge of the reference clock and a rising edge of the local clock. Note that this description is valid only when the PLL is close to synchronization: however, this assumption is not restrictive for studying the stability of the synchronized state, since we are only interested in small perturbations of the synchronized state. In practice, the DPD has a saturating characteristic, which ensures that the PLL behaves as a bang-bang PLL far from synchronization and, thus, has a wide lock-in range. A detailed description of these building blocks can be found in [17]. For the study of the single SS-ADPLL, it is supposed that the input signal comes from a regular reference clock.

A. Governing equations

Let $t_r[n]$ and $t_i[n]$ designate the time at which the nth rising edge of the reference clock and the DCO output respectively happen. We can write:

$$t_r[n+1] = t_r[n] + T_r\,, \tag{1}$$

where T_r is the period of the reference clock, and, knowing that the output of the PI filter $y_i[n]$ is updated at time $t_i[n]$:

$$t_i[n+1] = t_i[n] + T_i + g \cdot y_i[n] \tag{2}$$

where T_i is the period of the DCO internal clock and g is a multiplicative coefficient. Note that, in the present case, it is the period of the DCO which is controlled, not its frequency. Indeed, although modeling a PLL as a frequency controlled oscillator is the traditional way to proceed, the analysis presented here is carried out completely in the time domain, allowing an easy mathematical modeling of all the blocks in the real SS-ADPLL to be implemented for the HODISS project.

Now, it is clear that if $t_i[n] < t_r[n]$ (i.e. if the local clock is leading), the timing error

$$e_{ri}[n] = t_r[n] - t_i[n] \tag{3}$$

cannot be known at time $t_i[n]$ (Fig. 2). In order to update the value of $y_i[n]$ at every rising edge of the local clock, it is then necessary to provide the PI filter with an estimation $\hat{e}_{ri}[n]$ of $e_{ri}[n]$, whenever the local clock is leading. Throughout this study, it is assumed that

$$\hat{e}_{ri}[n] = e_{ri}[n-1], \tag{4}$$

which is possible since $e_{ri}[n-1]$ is always known at time $t_i[n]$. Depending on the practical implementation of the filter (see Appendix A), the control quantity $y_i[n]$ is either governed by:

$$y_i[n] = y_i[n-1] + C_1\varepsilon_{ri}[n] + C_2 e_{ri}[n-1] \tag{5-a}$$

or

$$y_i[n] = y_i[n-1] + C_1\varepsilon_{ri}[n] + C_2\varepsilon[n-1], \tag{5-b}$$

where $\varepsilon_{ri}[n] = \begin{cases} e_{ri}[n] & \text{if } e_{ri}[n] \le 0 \\ \hat{e}_{ri}[n] & \text{otherwise} \end{cases}$ (6)

and C_1 and C_2 are the filter coefficients. Equation (5-a) corresponds to the implementation shown in Fig. A2-a (filter type I), (5-b) corresponds to the implementation shown in Fig. A2-b (filter type II). Note that in an ideal PI filter, not accounting for self-sampling, the control quantity would be governed by:

$$y_i[n] = y_i[n-1] + C_1 e_{ri}[n] + C_2 e_{ri}[n-1]. \tag{5-c}$$

From (1) and (2), we have:

$$e_{ri}[n+1] = e_{ri}[n] + T_r - T_i - g \cdot y_i[n] \tag{7}$$

and one may then eliminate y_i from (5-a) to obtain:

$$e_{ri}[n+1] - 2e_{ri}[n] + e_{ri}[n-1] = -K_1\varepsilon_{ri}[n] - K_2 e_{ri}[n-1], \tag{8-a}$$

or from (5-b) to obtain:

$$e_{ri}[n+1] - 2e_{ri}[n] + e_{ri}[n-1] = -K_1\varepsilon_{ri}[n] - K_2\varepsilon_{ri}[n-1] \tag{8-b}$$

where $K_1 = g \cdot C_1$ and $K_2 = g \cdot C_2$. Note that (8-a) (resp. (8-b)) may be recast as two (resp. four) separate linear equations where only $e_{ri}[n+1]$ and its past values appear, each equation corresponding to a possible value of $\varepsilon_{ri}[n]$ and $\varepsilon_{ri}[n-1]$. These autonomous equations may also be rewritten in the classical state-space PLS form:

$$\mathbf{x}[n+1] = \mathbf{A}_k \mathbf{x}[n] \tag{9}$$

where $\mathbf{x}[n] = [e_{ri}[n] \quad e_{ri}[n-1] \quad e_{ri}[n-2]]^T$ and $\mathbf{A}_k$ is the state matrix of the kth cell of state-space. The transition diagrams from cell to cell are represented in Fig. 3 for both types of filter.

The synchronization of type I and type II SS-ADPLLs has been addressed in [8-9]. Sub-sections II-B and II-C give a summary of the main results that can be found in these papers.

B. Synchronization analysis for a single SS-ADPLL

A single SS-ADPLL synchronizes when $e_{ri}[n]$ goes to zero or, equivalently, when the piecewise-linear systems (PLSs) defined by (8-a) or (8-b) are asymptotically stable. Thus, it is of interest for the designer to determine the stability domain of the PLL, defined as the region of the (K_1, K_2) plane for which (8-a) or (8-b) is stable.

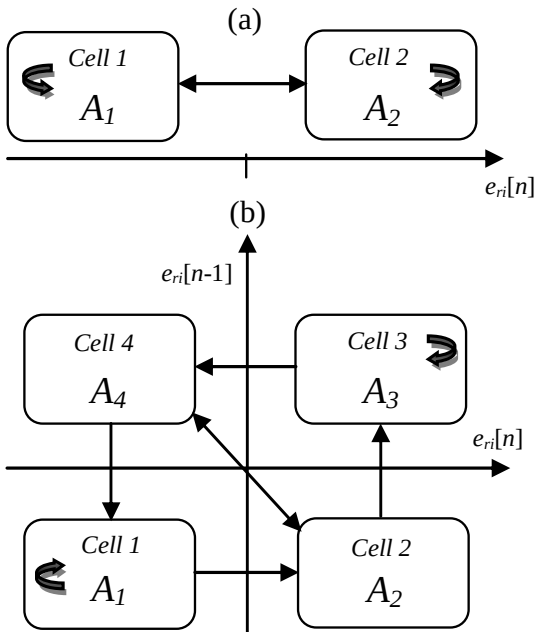


Fig. 3. Transition diagrams of type I (3-a) and type II (3-b) SS-ADPLLs.

1) Type I SS-ADPLLs

In [8], the following sufficient stability conditions on K_1 and K_2 were established for (8-a):

$$\begin{cases} K_1 + K_2 > 0 \\ K_1^{\,2} - 4(K_1 + K_2) > 0 \\ 1 + K_2 > 0 \\ 0 < K_1 < 2 \end{cases}. \tag{10}$$

However, transient simulations showed that these conditions, based on analytical considerations, were rather conservative. Another approach, based on the averaging of (8-a), led to an over-estimation of the stability domain. More precisely, a so-called "average system" was defined by replacing $\varepsilon_{ri}[n]$ in (8-a) by:

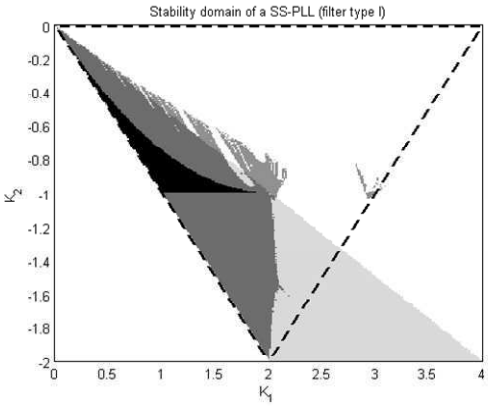
$$\tilde{\varepsilon}_{ri}[n] = \frac{1}{2}\left(e_{ri}[n] + e_{ri}[n-1]\right). \tag{11}$$

The resulting system is linear and its stability can be assessed from the position of its poles. The stability domains obtained with the three approaches are represented in Fig. 4-a.

Note that the stability of a PLS with one (or more) unstable cell cannot be assessed with PQLFs, unless, by construction, the PLS leaves the unstable cell(s) immediately after entering it [9,14-15]. Hence PQLFs cannot be used to determine the stability domain of type I SS-ADPLLs, because the equation governing the system in cell 2 (Fig. 3-a), corresponding to $e_{ri}[n] > 0$, is unstable and there is no guarantee that the system will bounce back into cell 1 immediately after entering

cell 2. General results concerning the use of PQLFs are summed up in Appendix B.

(a)



(b)

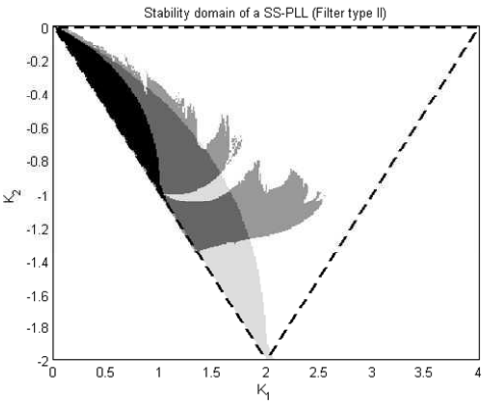


Fig. 4. Stability domain of a single type I (4-a) or type II (4-b) SS-ADPLL. The dark grey areas are obtained by transient simulations and the light gray areas are derived by calculating the poles of the average systems. In (4-a), the black area is the one defined by (10) whereas in (4-b), it is calculated through PQLFs. The dashed lines are the limits of the stability domain of an ideal ADPLL governed by (5-c).

2) Type II SS-ADPLLs

The study of the stability of a single type II SS-ADPLL is more involved than that of a type I SS-ADPLL: we have been unable to derive analytical bounds for the filter coefficients. However, PQLFs can be used to derive sufficient stability conditions for type II SS-ADPLLs: although the equations of cell 2 and cell 4 can be shown to be unstable, the system leaves these two cells immediately after entering them.

The stability domains of type II SS-ADPLLs determined by PQLFs (as explained in Appendix C), by transient simulation and by averaging are represented in Fig. 4-b. Once more, we find that the sufficient stability conditions established through PQLFs are rather conservative, whereas the averaging approach leads to over-estimate the stability domain of the system.

C. Synchronization analysis for an autonomous network of two SS-ADPLLs

Although the results concerning the synchronization of single type I or type II SS-ADPLLs with an outside reference clock are rather inconclusive, we showed in [8-9] that small autonomous networks of two identical SS-ADPLLs are much simpler to study. For example, suppose the reference clock is replaced by another SS-ADPLL of the same type (say type I), as shown in Fig. 5. One of the PLLs is then governed by (8-a), whereas the other is governed by:

$$e_{ir}[n+1]-2e_{ir}[n]+e_{ir}[n-1]=-K_1\varepsilon_{ir}[n]-K_2e_{ir}[n-1], \qquad (12)$$

where the same conventions as above are used. Now it is clear that:

$$e_{ri}[n]=-e_{ir}[n]\ , \qquad (13)$$

and also, from (6), that:

$$\varepsilon_{ri}[n]-\varepsilon_{ir}[n]=e_{ri}[n]+e_{ri}[n-1]. \qquad (14)$$

Thus, subtracting (12) from (8-a) yields a single linear equation governing e_{ri}:

$$e_{ri}[n+1]-2e_{ri}[n]+e_{ri}[n-1]=-\frac{K_1}{2}e_{ri}[n]-\left(K_2+\frac{K_1}{2}\right)e_{ri}[n-1] \qquad (15)$$

It is then very simple to determine the roots of the characteristic polynomial of (15) and establish the conditions under which this small network synchronizes.

We show in section III that, in some way, this remarkable linearization property can be extended to Cartesian networks of arbitrary size.

III. Synchronization of Cartesian Networks of SS-ADPLLs of Arbitrary size

1) Framework

An autonomous Cartesian network of SS-ADPLLs consists in a rectangular (two-dimensional) grid of self-sampled nodes, each node being connected to at most 4 neighbors. A typical network and node are represented in Fig. 6. The output of the DPD of the kth node on the nth rising edge of the local clock $t_k[n]$ is equal to:

$$\varepsilon_k[n]=\frac{1}{|V_k|}\sum_{l\in V_k}\varepsilon_{lk}[n], \qquad (16)$$

with

$$\varepsilon_{lk}[n]=\begin{cases} e_{lk}[n] & \text{if } e_{lk}[n]\le 0 \\ e_{lk}[n-1] & \text{otherwise} \end{cases}, \qquad (17)$$

$$e_{lk}[n]=t_l[n]-t_k[n]=-e_{kl}[n], \qquad (18)$$

where V_k is the set of the indices of the nodes in a neighborhood of the kth node and $|V_k|$ is the cardinal of V_k, i.e. $|V_k|$ is equal to 2 for corner nodes, 3 for edge nodes and 4 otherwise. From (17) and (18), the following fundamental equality can be derived:

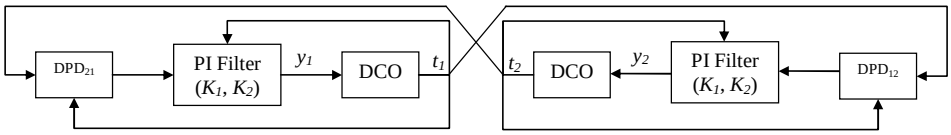


Fig. 5. An autonomous network of two SS-ADPLLs

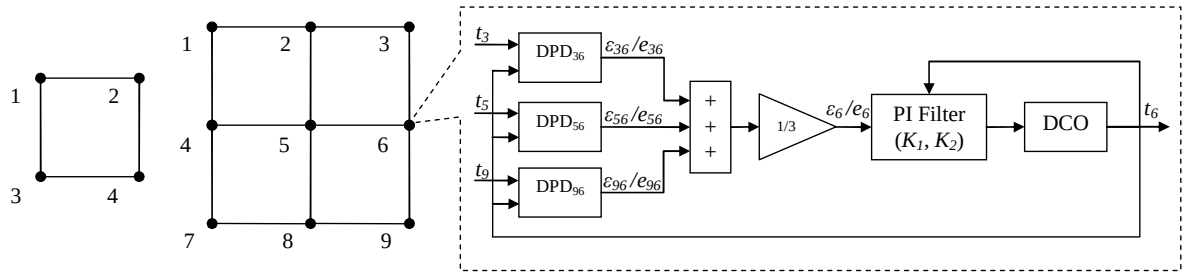


Fig. 6. Two Cartesian networks of 4 or 9 SS-ADPLLs. At each node, the total error is defined as the mean value of the outputs of the DPDs.

$$\varepsilon_{lk}[n] - \varepsilon_{kl}[n] = e_{lk}[n] + e_{lk}[n-1]. \tag{19}$$

Let us also define $\boldsymbol{\varepsilon}[n]$ a vector whose k^{th} coordinate is $\varepsilon_k[n]$ and $\mathbf{e}[n]$, a vector whose k^{th} coordinate is

$$e_k[n] = \frac{1}{|V_k|}\sum_{l\in V_k} e_{lk}[n]. \tag{20}$$

Each SS-ADPLL uses $\varepsilon_k[n]$ (16) to update the local filter output at time $t_k[n]$, as in section II. One may then assemble the equations governing the whole network and find, for type I SS-ADPLLs:

$$\mathbf{e}[n+1] - 2\mathbf{e}[n] + \mathbf{e}[n-1] = -\mathbf{L}\left(K_1\boldsymbol{\varepsilon}[n] + K_2\mathbf{e}[n-1]\right), \tag{21-a}$$

and, for type II SS-ADPLLs:

$$\mathbf{e}[n+1] - 2\mathbf{e}[n] + \mathbf{e}[n-1] = -\mathbf{L}\left(K_1\boldsymbol{\varepsilon}[n] + K_2\boldsymbol{\varepsilon}[n-1]\right) \tag{21-b}$$

where $\mathbf{L}$ is the normalized Laplacian matrix of the network, defined as:

$$L_{kl} = \begin{cases} 1 & \text{if } k = l \\ -\dfrac{1}{|V_k|} & \text{if } l \in V_k \\ 0 & \text{otherwise} \end{cases}. \tag{22}$$

More generally, given an arbitrary loop filter with one pole in zero (in order to ensure phase synchronization), it will always be possible to write the equations governing the network in the form:

$$\mathbf{e}[n+1] - 2\mathbf{e}[n] + \mathbf{e}[n-1] = \mathbf{L}\left(\sum_{p=0}^{P} K_p\boldsymbol{\varepsilon}[n-p] + \sum_{q=0}^{Q} K_q^{'}\mathbf{e}[n-q]\right). \tag{23}$$

For a network of I lines and J columns, a node number k being given, we can unambiguously define the line index i_k and the column index j_k so that

$$k = (i_k - 1)J + j_k \tag{24}$$

where $i_k \in \{1,\ldots,I\}$, $j_k \in \{1,\ldots,J\}$.

Let us define a vector $\mathbf{v}$ of size IJ, such that:

$$v_k = (-1)^{i_k + j_k}|V_k|. \tag{25}$$

For example, for the 3-by-3 network of Fig. 6, we have:

$$\mathbf{v}^T = [2 \quad -3 \quad 2 \quad -3 \quad 4 \quad -3 \quad 2 \quad -3 \quad 2]. \tag{26}$$

Finally, we introduce the "master equation" of the network governing the quantity:

$$E[n] = \mathbf{v}^T\mathbf{e}[n]. \tag{27}$$

as:

Definition 1 - *The master equation of the network is the scalar equation obtained by projecting the governing equations of the network ((21-a), (21-b) and more generally (23)) on* $\mathbf{v}$ *(25).*

This master equation is a restriction of the original PLS to a one-dimensional subspace, to which it stays confined if the initial state vector $\mathbf{e}[0]$ is collinear to $\mathbf{v}$. Hence, it is clearly necessary that the master equation be stable for the whole PLS to be stable.

Now, we shall prove that, for a network of SS-ADPLLs, the master equation is in fact *linear*. It is then straightforward to find a necessary condition for the stability of the whole network, by computing the roots of its characteristic polynomial. We shall then demonstrate the following theorem:

Theorem 1 - *The master equation of an autonomous Cartesian network of identical SS-ADPLLs is linear. A*

necessary condition for the stability and the synchronization of the network is then that the roots of the characteristic polynomial of the master equation are within the unit circle.

2) Proof of Theorem 1

The proof is organized as follows. First, we prove the following property:

Property 1 - $\mathbf{v}^T$ *is the left eigenvector of the normalized Laplacian matrix associated to the eigenvalue* $\lambda = 2$.

Then, we establish:

Property 2 - *In an autonomous Cartesian network of identical SS-ADPLLs verifying (19), the following equality holds:*

$$\mathbf{v}^T \boldsymbol{\varepsilon}[n] = \frac{1}{2}\left(E[n] + E[n-1]\right).$$

If Properties 1 and 2 hold, the projection of (21-a), (21-b) or, more generally, (23) on $\mathbf{v}$ then results in a linear equation governing $E[n]$, from which a necessary stability condition can be inferred, as explained in sub-section III-1, thus proving Theorem 1.

a) Proof of Property1

Let $\mathbf{u}^T = \mathbf{v}^T \mathbf{L}$. We have:

$$u_l = \sum_{k=1}^{IJ} v_k L_{kl} = \sum_{k=1}^{IJ} (-1)^{i_k + j_k} |V_k| L_{kl}. \tag{28}$$

From (22), the only non-vanishing terms under the sum sign correspond to the values $k = l$ or $k / l \in V_k$. Now we note that, in a Cartesian network, if $l \in V_k$, we are in one of the following to cases:

$$i_l = i_k \pm 1$$

or

$$j_l = j_k \pm 1.$$

Hence we find that, if $l \in V_k$:

$$(-1)^{i_l + j_l} = -(-1)^{i_k + j_k}. \tag{29}$$

Thus (28) becomes:

$$\begin{aligned} u_l &= (-1)^{i_l + j_l} |V_l| - \sum_{\substack{k=1 \\ k/l \in V_k}}^{IJ} (-1)^{i_k + j_k} \\ &= (-1)^{i_l + j_l} \left(|V_l| + \sum_{\substack{k=1 \\ k/l \in V_k}}^{IJ} 1 \right) \end{aligned}. \tag{30}$$

Now the last term on the right-hand side is the cardinal of V_l, so that we have:

$$u_l = 2(-1)^{i_l + j_l} |V_l| = 2v_l. \tag{31}$$

This completes the first step of the proof. It is notable that the eigenvalues of the normalized Laplacian matrix of a Cartesian network are necessarily inferior to 2 [18]. Thus, $\mathbf{v}^T$ is the eigenvector associated to the largest eigenvalue of the normalized Laplacian.

b) Proof of Property 2

The second step of the proof is straightforward, but rather tedious. First, we shall show that:

$$S = \sum_{k=1}^{IJ} (-1)^{i_k + j_k} \sum_{l \in V_k} \varepsilon_{lk}[n] + \varepsilon_{kl}[n] = 0. \tag{32}$$

Let L_k (resp. C_k) be the set of the indices of the nodes in the neighborhood and on the same line (resp. column) as node k. It is clear that:

$$V_k = L_k \cup C_k. \tag{33}$$

Thus, S may be rewritten:

$$\begin{aligned} S &= \sum_{k=1}^{IJ} (-1)^{i_k + j_k} \sum_{l \in L_k} \varepsilon_{lk}[n] + \varepsilon_{kl}[n] \\ &+ \sum_{k=1}^{IJ} (-1)^{i_k + j_k} \sum_{l \in C_k} \varepsilon_{lk}[n] + \varepsilon_{kl}[n] \end{aligned}. \tag{34}$$

Letting S_L (resp. S_C) be the sum over the lines (resp. columns), i.e. the first (resp. second) sum on the right-hand side of (34), we have:

$$\begin{aligned} S_L &= \sum_{k=1}^{IJ} (-1)^{i_k + j_k} \sum_{l \in L_k} \varepsilon_{lk}[n] + \varepsilon_{kl}[n] \\ &= \sum_{i=1}^{I} \sum_{j=1}^{J} (-1)^{i+j} \sum_{l \in L_{(i-1)J+j}} \varepsilon_{l,(i-1)J+j}[n] + \varepsilon_{(i-1)J+j,l}[n] \\ &= \sum_{i=1}^{I} (-1)^i \sum_{j=1}^{J} \sum_{l \in L_{(i-1)J+j}} (-1)^j \left(\varepsilon_{l,(i-1)J+j}[n] + \varepsilon_{(i-1)J+j,l}[n] \right) \\ &= \sum_{i=1}^{I} (-1)^i \Theta_i \end{aligned} \tag{35}$$

Now, one may expand Θ_i:

$$\begin{aligned} \Theta_i = & \\ & -\left(\varepsilon_{(i-1)J+2,(i-1)J+1}[n] + \varepsilon_{(i-1)J+1,(i-1)J+2}[n]\right)\} j = 1 \\ & \left.\begin{aligned} &+\left(\varepsilon_{(i-1)J+2,(i-1)J+1}[n] + \varepsilon_{(i-1)J+1,(i-1)J+2}[n]\right) \\ &+\left(\varepsilon_{(i-1)J+3,(i-1)J+2}[n] + \varepsilon_{(i-1)J+2,(i-1)J+3}[n]\right) \end{aligned}\right\} j = 2 \\ & \left.\begin{aligned} &-\left(\varepsilon_{(i-1)J+3,(i-1)J+2}[n] + \varepsilon_{(i-1)J+2,(i-1)J+3}[n]\right) \\ &-\left(\varepsilon_{(i-1)J+4,(i-1)J+3}[n] + \varepsilon_{(i-1)J+3,(i-1)J+4}[n]\right) \end{aligned}\right\} j = 3 \\ & + \ldots \\ & \left.\begin{aligned} &+(-1)^{J-1}\left(\varepsilon_{(i-1)J+J-2,(i-1)J+J-1}[n] + \varepsilon_{(i-1)J+J-1,(i-1)J+J-2}[n]\right) \\ &+(-1)^{J-1}\left(\varepsilon_{(i-1)J+J,(i-1)J+J-1}[n] + \varepsilon_{(i-1)J+J-1,(i-1)J+J}[n]\right) \end{aligned}\right\} j = J-1 \\ & +(-1)^{J}\left(\varepsilon_{(i-1)J+J,(i-1)J+J-1}[n] + \varepsilon_{(i-1)J+J-1,(i-1)J+J}[n]\right)\} j = J \end{aligned} \tag{36}$$

and find that every line of (36) is cancelled out by the following one. Hence, $\Theta_i = 0$ and $S_L = 0$. Similarly, one may show that $S_C = 0$, which proves (32).

Now, we may complete the rest of the proof by writing:

$$\begin{aligned}\mathbf{v}^T\boldsymbol{\varepsilon}[n] &= \sum_{k=1}^{IJ}(-1)^{i_k+j_k}|V_k|\varepsilon_k[n] \\ &= \sum_{k=1}^{IJ}(-1)^{i_k+j_k}\sum_{l\in V_k}\varepsilon_{lk}[n]\end{aligned}. \tag{37}$$

Using (32), we may rewrite (37) as:

$$\mathbf{v}^T\boldsymbol{\varepsilon}[n] = \frac{1}{2}\sum_{k=1}^{IJ}(-1)^{i_k+j_k}\sum_{l\in V_k}\varepsilon_{lk}[n]-\varepsilon_{kl}[n], \tag{38}$$

which simplifies, using (19) ,into:

$$\begin{aligned}\mathbf{v}^T\boldsymbol{\varepsilon}[n] &= \frac{1}{2}\sum_{k=1}^{IJ}(-1)^{i_k+j_k}\sum_{l\in V_k}e_{lk}[n]+e_{lk}[n-1] \\ &= \frac{1}{2}\sum_{k=1}^{IJ}(-1)^{i_k+j_k}|V_k|\left(\frac{1}{|V_k|}\sum_{l\in V_k}e_{lk}[n]+e_{lk}[n-1]\right) \\ &= \frac{1}{2}\left(E[n]+E[n-1]\right)\end{aligned}. \tag{39}$$

This completes the second part of the proof.

3) Discussion

Before illustrating these results, we stress the fact that the master equations obtained by projection of the network equations on $\mathbf{v}$ are independent of the number of lines and columns in the network. They only depend on the transfer function of the loop filter and its practical implementation.

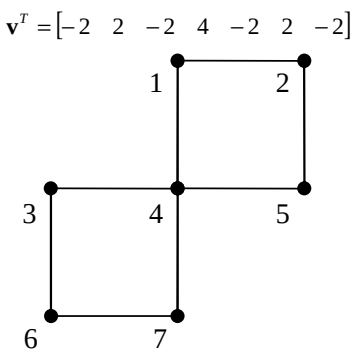


Fig. 7. An incomplete Cartesian network of 7 SS-ADPLLs with the corresponding vector $\mathbf{v}^T$.

Furthermore, it is quite simple to extend Theorem 1 to incomplete Cartesian networks, i.e. Cartesian network with one or more nodes missing (Fig. 7) and thus to establish the following corollary:

Corollary 1 - *Theorem 1 holds for any incomplete autonomous Cartesian network of identical SS-ADPLLs.*

Properties 1 and 2 are unchanged by the incompleteness of the network, the only difference being in the proof of Property 2, where one must pay attention to the presence of "holes" in Θ_i.

Finally, suppose we have chosen to predict the timing error in a more elaborate way, according to:

$$\varepsilon_{lk}[n] = \begin{cases} e_{lk}[n] & \text{if } e_{lk}[n]\le 0 \\ \sum_{i=1}^{I} a_i e_{lk}[n-i] & \text{otherwise} \end{cases}. \tag{40}$$

Fundamental equality (19) then becomes:

$$\varepsilon_{lk}[n]-\varepsilon_{kl}[n] = e_{lk}[n]+\sum_{i=1}^{I}a_i e_{lk}[n-i], \tag{41}$$

It is then possible to derive a master equation using Property 1 and the following property:

Property 3 - *In an autonomous Cartesian network of identical SS-ADPLLs verifying (41), the following equality holds:*

$$\mathbf{v}^T\boldsymbol{\varepsilon}[n] = \frac{1}{2}\left(E[n]+\sum_{i=1}^{I}a_i E[n-i]\right).$$

The proof of Property 3 is almost the same as that of Property 2, except it is (41) (as opposed to (19)) which is injected into (38), the previous part of the proof being independent of which fundamental equality is verified.

IV. Illustrations and Results

A. Master equations of type I and type II SS-ADPLLs

Let us, as an illustration, derive the master equations of type I and type II SS-ADPLLs. The projection of (21-a) on $\mathbf{v}$ leads to:

$$E[n+1]-2E[n]+E[n-1] = -K_1\mathbf{v}^T\mathbf{L}\boldsymbol{\varepsilon}[n]-K_2\mathbf{v}^T\mathbf{L}\mathbf{e}[n-1]. \tag{42}$$

Using Property 1, (42) becomes:

$$E[n+1]-2E[n]+E[n-1] = -2K_1\mathbf{v}^T\boldsymbol{\varepsilon}[n]-2K_2E[n-1]. \tag{43}$$

Using Property 2, the master equation of type I SS-ADPLLs can then be derived from (43):

$$\begin{aligned}&E[n+1]-2E[n]+E[n-1]= \\ &-K_1E[n]-(K_1+2K_2)E[n-1]\end{aligned} \tag{44-a}$$

Similarly, for type II SS-ADPLLs, we obtain:

$$\begin{aligned}&E[n+1]-2E[n]+E[n-1]= \\ &-K_1E[n]-(K_1+K_2)E[n-1]-K_2E[n-2]\end{aligned} \tag{44-b}$$

Incidentally, it is always possible to find a network of ideal ADPLLs, i.e. which are not self-sampled, with the same topology as the original one, whose master equation is the same as that of the self-sampled network. For example, (44-a) and (44-b) are also the master equations (i.e. the projections on $\mathbf{v}$) of the networks governed by:

$$\begin{aligned}&\mathbf{e}[n+1]-2\mathbf{e}[n]+\mathbf{e}[n-1]= \\ &-\mathbf{L}\left(\frac{K_1}{2}\mathbf{e}[n]+\left(K_2+\frac{K_1}{2}\right)\mathbf{e}[n-1]\right)\end{aligned} \tag{45-a}$$

and

$$\mathbf{e}[n+1]-2\mathbf{e}[n]+\mathbf{e}[n-1]= \\ -\mathbf{L}\left(\frac{K_1}{2}\mathbf{e}[n]+\left(\frac{K_1+K_2}{2}\right)\mathbf{e}[n-1]+\frac{K_2}{2}\mathbf{e}[n-2]\right). \quad (45\text{-b})$$

It is also interesting to notice that the resulting ideal networks are in fact the "average networks", as defined in sub-section II-B-1, which can be obtained from (21-a) or (21-b) by replacing in these equations $\boldsymbol{\varepsilon}[n]$ by:

$$\tilde{\boldsymbol{\varepsilon}}[n]=\frac{1}{2}\left(\mathbf{e}[n]+\mathbf{e}[n-1]\right). \quad (46)$$

Whether some properties of the original self-sampled network can be derived from those of the associated "average network" remains to be demonstrated.

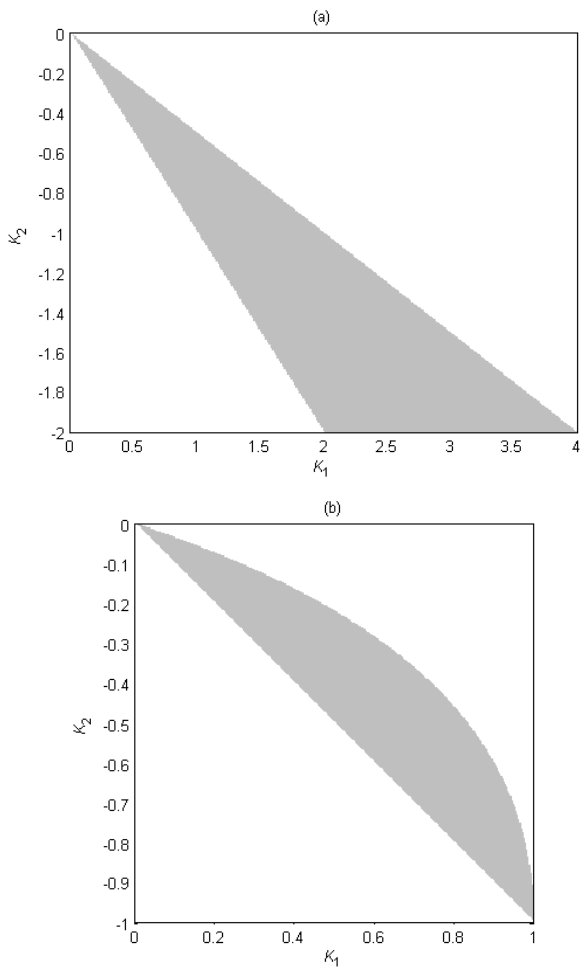


Fig. 8. Stability domain (grey area) of the master equation of networks of type I (8-a) or type II (8-b) SS-ADPLLs, determined by finding the roots of the characteristic polynomials of (44-a) and (44-b).

B. General master equation

More generally, assuming the general form (23) and timing error prediction (40), the master equation is found to be:

$$E[n+1]-2E[n]+E[n-1]= \\ \sum_{p=0}^{P}K_p E[n-p]+2\sum_{q=0}^{Q}K_q^{'}E[n-q]+\sum_{p=0}^{P}\sum_{i=1}^{I}a_i K_p E[n-p-i]. \quad (47)$$

As in IV-A, the corresponding "average network" could be obtained by substituting in (23) $\boldsymbol{\varepsilon}[n]$ by:

$$\tilde{\boldsymbol{\varepsilon}}[n]=\frac{1}{2}\left(\mathbf{e}[n]+\sum_{i=1}^{I}a_i\mathbf{e}[n-i]\right). \quad (48)$$

C. Stability domains of autonomous networks of type I and type II SS-ADPLLs

The stability domains derived from the characteristic polynomials of (44-a) and (44-b) are represented in Fig. 8. It is remarkable that the stability domains derived from the transient simulation of (21-b) or from solving the LMIs of Appendix B yield exactly the same results, regardless of the network size. This suggests that the stability of (44-b) is a necessary *and sufficient* condition for the synchronization of autonomous Cartesian networks of type II SS-ADPLLs. Furthermore, it is simple to verify that the average network (45-b) also has the same stability domain.

On the other hand, the results obtained with type I SS-ADPLLs from simulation depend on the size of the network: for small networks of 2 or 4 SS-ADPLLs, the stability domain given by simulation or PQLFs does indeed coincide with the one derived from (44-a). However, as the network size increases, the actual stability domain becomes smaller than predicted with the master equation (Fig. 9). The Matlab code used for the transient simulation of the different networks is given in Appendix D. If the norm of E_N_MINUS_1 is small after a sufficient number of edges (depending on the size of the network and the values of K_1 and K_2), the network is assumed to be stable.

D. Transient behaviour of networks of type I and type II SS-ADPLLs

Transient simulations of networks of 4 SS-ADPLLs (as depicted in Fig. 6) are now performed. First, a network composed of type I SS-ADPLLs is simulated. The coefficients of the PI filter are chosen as $K_1=1.6$ and $K_2=-1.4$, in order to enforce a strong correction while remaining stable. The total error of each node, calculated from (21-a) is represented in Fig. 10, along with that of the "average network" (45-a), launched from the same initial conditions. The results show that, although the two networks seem to synchronize in the same amount of time, their transient behaviour is quite dissimilar. As K_1 and K_2 decrease, the responses of both networks become more alike (but their settling time increases).

On the other hand, the response of a network of type II SS-ADPLLs (21-b) is very similar to that of the corresponding "average network" (45-b), even for large values of K_1 and K_2. A typical response is plotted in Fig. 11, for $K_1=0.8$ and $K_2=-0.7$.

These observations and those of sub-section IV-C lead us to the conclusion that the average network is a good basis for the design of networks of type II SS-ADPLLs. Its behaviour is

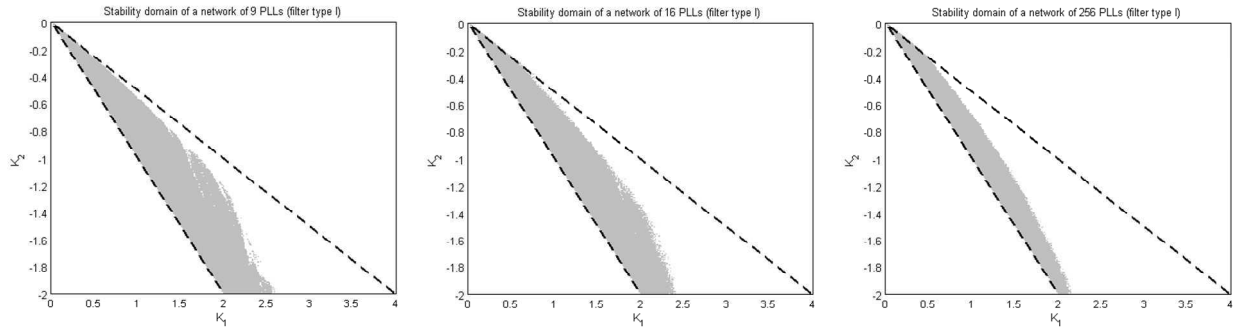


Fig. 9 - Stability domains of complete Cartesian networks of (from left to right) of 9, 16 or 256 type I SS-ADPLLs. All networks have the same number of lines and columns. The dashed lines represent the limits of the stability domain derived from the master equation (44-a).

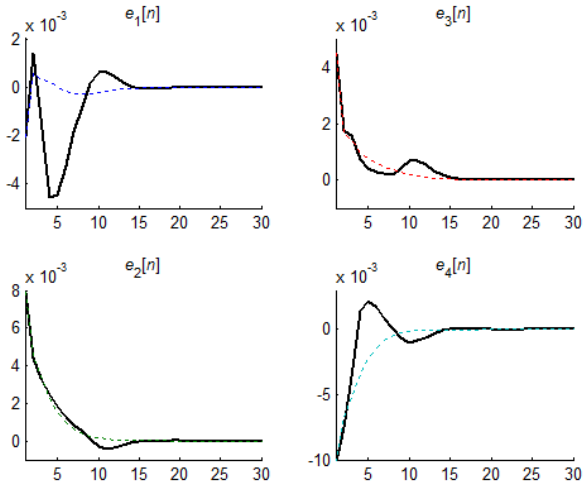


Fig. 10 - Total errors in a network of 4 type I SS-ADPLLs (black bold lines) compared with the total errors of the corresponding average system (colored dashed lines).

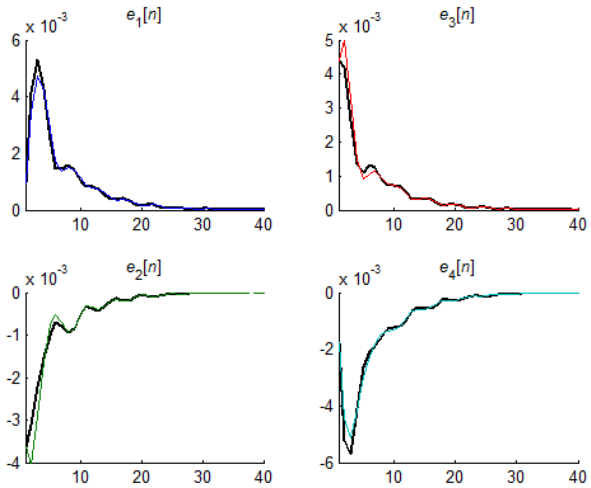


Fig. 11. Total errors in a network of 4 type II SS-ADPLLs (black bold lines) compared with the total errors of the corresponding average system (colored dashed lines).

very close to that of the self-sampled network (same stability domain, similar transient behaviour) and it has the great advantage of being linear. Thus, all the classical tools from linear analysis (stability margins, for example) can be used as a first step in the design, and their results tested afterwards with transient simulations of the actual SS-ADPLL networks.

V. Conclusion

The study of the stability of a clock distribution network based on SS-ADPLLs was described in this paper. We showed that (networks of) SS-ADPLLs can be modeled as piecewise-linear systems. Three approaches were tested in order to determine their stability (or synchronization) domains. The most rigorous approach, based on PQLFs, is quite costly in computing effort and requires a huge amount of bookkeeping, even for networks of moderate size. Moreover, it cannot be applied as is to all sorts of SS-ADPLLs and yields only sufficient stability conditions. The most straightforward approach, based on transient simulation of the network, yields some results which may be dependent on the initial conditions of the system. The third approach, which is the main contribution of this paper, may seem quite limited: it relies on a particular network topology (Cartesian), is valid for autonomous networks only and it yields necessary (not sufficient) stability conditions. However, it does provide us with a very simple tool (the so-called master equation) to determine, as a starting point, the limits of the domain in which the filter coefficients must be chosen to ensure stability. Furthermore, our study indicates that the properties (not only stability, but also settling time) of Cartesian networks of type-II SS-ADPLLs can safely be derived from those of the corresponding average network. A rigorous demonstration of this property remains yet to be established.

The study of non-autonomous networks of SS-ADPLLs and of the influence of the nonlinearity of the characteristic of the DPD is the subject of ongoing work.

Acknowledgment

This work is supported by the French National Agency of Research (ANR) through the HODISS project.

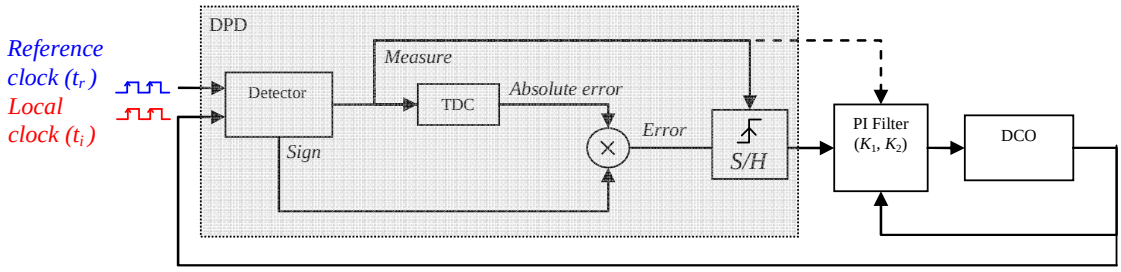


Fig. A1. Detailed block-diagram of an SS-ADPLL.

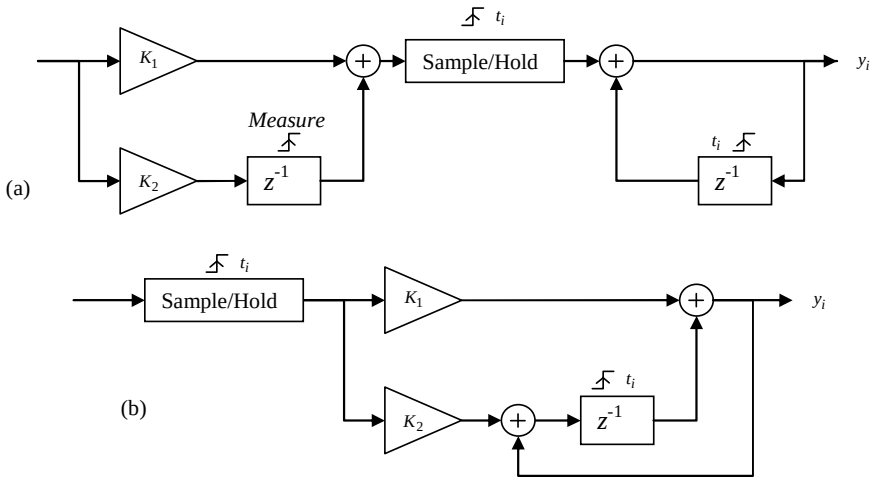


Fig. A2 - Implementation of type I (a) and type II (b) PI filters.

REFERENCES

[1] G.A. Pratt, J. Nguyen, "Distributed Synchronous Clocking", IEEE Transactions on Parallel and Distributed System, vol. 6, pp. 314-28, 1995.

[2] V. Gutnik, A. P. Chandrakasan, "Active GHz Clock Network Using Distributed PLLs", IEEE Journal of Solid-State Circuits, vol. 35, pp. 1553-60, 2000.

[3] E.G. Friedman, "Design and Analysis of a Hierarchical Clock Distribution System for Synchronous Standard Cell/Macrocell VLSI", IEEE Journal of Solid-State Circuits, vol. 21, pp. 240-246, 1986.

[4] M. Saint-Laurent, M. Swaminathan, " A Multi-PLL Clock Distribution Architecture for Gigascale Integration", IEEE Computer Society Workshop on VLSI, pp. 30-35, 2001.

[5] S.H. Strogatz, "Sync: The Emerging Science of Spontaneous Order", Hyperion, New York, 2003.

[6] W. C. Lindsey et al., "Network Synchronization," Proceedings of the IEEE, vol. 73, no.10, Oct. 1985, pp. 1445–67.

[7] O. Simeone et al., "Distributed synchronization in wireless networks," Proc. IEEE, vol. 25, Sept. 2008, pp. 81–97.

[8] J-M. Akré, J. Juillard, D. Galayko, and E. Colinet "Synchronized State in Networks of Digital Phase-Locked Loops", 8th IEEE International NEWCAS Conference, pp. 89-92, Montreal, 2010.

[9] J-M. Akré, J. Juillard, S. Olaru, D. Galayko, and E. Colinet "Determination of the Behaviour of Self-Sampled Digital Phase-Locked Loops", 53rd IEEE International MWSCAS'10, pp. 1089-1092, Seattle, Washington (USA), August 1-4, 2010.

[10] R. Flynn, and O.Feely, "Limit Cycles in Digital Bang-Bang PLLs", 18th European Conference on Circuit Theory and Design (ECCTD), Aug. 2007, pp 731-734.

[11] R.C. Walker, "Designing Bang-Bang PLLs for Clock and Data Recovery in Serial Data Transmission Systems", Phase-Locking in High-Performance Systems, B. Razavi, Ed.:IEEE Press, pp 34-45, 2003.

[12] N. Da Dalt, "A design-oriented study of the nonlinear dynamics of digital bang-bang PLLs", IEEE Transactions on Circuits and Systems I, vol. 52, pp. 21-31, 2005.

[13] I. L. Syllaios, R. B. Staszewski, P. T. Balsara, "Time-Domain Modeling of an RF All-Digital PLL", IEEE Transactions on Circuits and Systems II, vol. 55, pp. 601-605, 2008.

[14] G. Feng, "Stability analysis of piecewise discrete-time linear systems", IEEE Transactions on Automatic Control, vol. 47, pp. 1108-12, 2002.

[15] M. Johansson, and A. Rantzer, "Computation of piecewise quadratic Lyapunov functions for hybrid systems," IEEE Transactions on Automatic Control, vol. 43, pp. 555–559, 1998.

[16] F. M. Gardner, "Phaselock Techniques", Wiley-Interscience, New York, 1979.

[17] E. Zianbetov et al., "Design and VHDL modeling of all-digital PLLs", 8th IEEE International NEWCAS Conference, pp. 293-296, Montreal, 2010.

[18] B. Mohar, "Some Applications of Laplace Eigenvalues of Graphs", in *Graph Symmetry: Algebraic Methods and Applications*, Kluwer, 1997, pp. 225-275.

APPENDIX A - IMPLEMENTATION OF TYPE I AND TYPE II SS-ADPLLS

A detailed schematic of an SS-ADPLL is shown in Fig. A1. The DPD consists in an edge detector and in a time-to-digital converter (TDC). The edge detector delivers two binary signals: *Measure* (equal to 0 when a measure is being performed, to 1 otherwise) and *Sign* (equal to 1 when the local

clock is leading, to 0 otherwise). The output (*Absolute error*) of the TDC is then multiplied by *Sign* to obtain the *Error* signal, which is updated when *Measure* goes to 1.

Type I and type II filters can then be implemented as shown in Fig. A2. Note that, in order to implement type I filters in a network of PLLs, the K_2 branch must be reproduced a number of times equal to the number of neighbouring nodes, which makes this solution less attractive. One should also note that in both cases the value of the output y_i changes only on the rising edges of the local clock. If this were not so (for example, if all clocked blocks were operated on edges of the *Measure* signal), (2) would no longer hold. The behaviour of the filter would be simplified, but that of the DCO would be much more complex.

APPENDIX B - PIECEWISE-QUADRATIC STABILITY OF PLSS

A classical approach to the determination of the stability of nonlinear systems is via Lyapunov functions. A Lyapunov function is a positive function of the states of a system whose value decreases along all the possible trajectories of the system. The existence of a Lyapunov function is a sufficient condition for proving the stability of an autonomous system. Except in the most trivial cases, there exists no generic method to construct or check for the existence of such a function. However, in the particular case of PLSs, the problem of finding a Lyapunov function can be broken down into several sub-problems.

A discrete-time PLS can be represented for its analysis by:

$$\mathbf{x}[n+1] = \mathbf{A}_i \mathbf{x}[n],\ \mathbf{x} \in S_i \tag{B-1}$$

where $\mathbf{x} \in R^n$ is the state of the system, $\{S_i\}_{i \in I} \subset R^n$ is a partition of the state-space in a number of closed polyhedral subspaces, I is the set of the indices of the subspaces and $\mathbf{A}_i$ the matrix of the ith local model of the system. Let us also define Ω the set representing all the possible transitions from one region to another, i.e.:

$$\Omega = \{i, j / \mathbf{x}[n] \in S_i, \mathbf{x}[n+1] \in S_j, j \neq i\} \tag{B-2}$$

In some cases, it is possible to prove the stability of PLSs by finding a common quadratic Lyapunov function (CQLF), i.e. a function $V(\mathbf{x}) = \mathbf{x}^T \mathbf{P} \mathbf{x}$, $\mathbf{P} = \mathbf{P}^T > 0$, such that

$$\mathbf{A}_i^T \mathbf{P} \mathbf{A}_i - \mathbf{P} < 0,\ \forall i \in I \tag{B-3}$$

Determining the existence of a CQLF can be done by solving the set (B-3) of linear matrix inequalities (LMIs), which can be achieved with software such as Matlab.

However, many PLSs are stable, even though no CQLF exists. It may then be possible to prove stability by constructing piecewise-quadratic Lyapunov functions [14-15], $V_i(\mathbf{x}) = \mathbf{x}^T \mathbf{P}_i \mathbf{x}$, $i \in I$, so that the following relaxed stability conditions:

$$\mathbf{A}_i^T \mathbf{P}_i \mathbf{A}_i - \mathbf{P}_i + \mathbf{M}_i < 0,\ \forall i \in I \tag{B-4}$$

are satisfied, where $\mathbf{M}_i$ is a matrix such that $\mathbf{x}^T \mathbf{M}_i \mathbf{x} \geq 0$, $\forall \mathbf{x} \in S_i$, which can be constructed as follows. Since the cells S_i are polyhedral, it is easy to build for each of them a matrix $\mathbf{E}_i$ such that $\mathbf{E}_i \mathbf{x}$ has non-negative entries $\forall \mathbf{x} \in S_i$. Then, for any positive matrix $\mathbf{U}_i$ (i.e. any matrix with non-negative entries):

$$\mathbf{x}^T \mathbf{E}_i^T \mathbf{U}_i \mathbf{E}_i \mathbf{x} \geq 0,\ \forall \mathbf{x} \in S_i \tag{B-5}$$

and $\mathbf{M}_i$ can then be chosen as $\mathbf{M}_i = \mathbf{E}_i^T \mathbf{U}_i \mathbf{E}_i$. The main result in Feng's work [14] applied to discrete-time PLSs is summarized in the following theorem.

Theorem (Feng)

Consider the discrete-time PLS (B-1). If there exist some symmetric matrices $\mathbf{P}_i$, $\mathbf{U}_i$, $\mathbf{W}_i$ and $\mathbf{Q}_{ij}$, $i, j \in I$ such that $\mathbf{U}_i$, $\mathbf{W}_i$, and $\mathbf{Q}_{ij}$ are positive and the following LMIs are respected:

$$0 < \mathbf{P}_i - \mathbf{E}_i^T \mathbf{U}_i \mathbf{E}_i\ ,\ i \in I\ , \tag{B-6-a}$$

$$\mathbf{A}_i^T \mathbf{P}_i \mathbf{A}_i - \mathbf{P}_i + \mathbf{E}_i^T \mathbf{W}_i \mathbf{E}_i < 0\ ,\ i \in I\ , \tag{B-6-b}$$

$$\mathbf{A}_i^T \mathbf{P}_j \mathbf{A}_i - \mathbf{P}_i + \mathbf{E}_i^T \mathbf{Q}_{ij} \mathbf{E}_i < 0\ ,\ \{i, j\} \in \Omega\ . \tag{B-6-c}$$

then the origin of the PLS is asymptotically stable. Moreover, the function:

$$V(\mathbf{x}) = \mathbf{x}^T \mathbf{P}_i \mathbf{x}\ ,\ \mathbf{x} \in S_i \tag{B-7}$$

is a Lyapunov function for the system. Qualitatively, (B-6-a) enforces the positiveness of $\mathbf{x}^T \mathbf{P}_i \mathbf{x}$ for $\mathbf{x} \in S_i$. Equation (B-6-b) guarantees that some energy is lost while the system resides in S_i. Finally, (B-6-c) ensures that some energy is also lost as the system moves from one cell to the other.

APPENDIX C - APPLICATION OF PQLF TO TYPE II SS-ADPLLS

A single type II SS-ADPLL is governed by (8-b), which can be rewritten in state-space form as (9), with:

$$\mathbf{A_1} = \begin{bmatrix} 2-K_1 & -1-K_2 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \tag{C-1-a}$$

$$\mathbf{A}_2 = \begin{bmatrix} 2 & -1-K_1-K_2 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \tag{C-1-b}$$

$$\mathbf{A}_3 = \begin{bmatrix} 2 & -(1+K_1) & -K_2 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \tag{C-1-c}$$

$$\mathbf{A}_4 = \begin{bmatrix} 2-K_1 & -1 & -K_2 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}. \tag{C-1-d}$$

For a given couple (K_1, K_2), one may try to solve the LMIs (B-6) and thus establish the stability of the system. Some

proper care must be taken to remove unfeasible LMIs. For example, $\mathbf{A}_2$ always has an eigenvalue whose modulus is larger than 1. Thus, it is impossible to satisfy (B-6-b) for $i = 2$. However, by construction, the system cannot reside in S_2 for more than one time step. Thus, the corresponding LMI does not have to be satisfied. The same goes for S_4. The remaining matrix inequalities can easily be solved with Matlab.

One of the main difficulties in using this approach resides in the bookkeeping effort that must be made to rule out unfeasible LMIs from the original set. As shown above, this is fairly simple in the case of a single SS-ADPLL. Unfortunately, the number of cells in the state-space increases exponentially with the size of the network, making it exceedingly hard to keep track of all possibilities, even for moderate network sizes.

APPENDIX D - MATLAB CODE FOR TRANSIENT SIMULATION

Define :

- N_EDGES, maximum number of rising edges (i.e. maximal simulation duration)
- N_NODES, number of nodes in the network
- LAP, normalized Laplacian $\mathbf{L}$ of the network
- K1 and K2, the tested values of K_1 and K_2
- PERIODS, column vector containing the values of the nominal periods of the DCOs
- T_START, column vector containing the moment of the first rising edge of each DCO
- TYPE, type of PLL, may be ’I’or ’II’ in the following code

Execute :

```
% Define network adjacency matrix ADJ
ADJ=eye(N_NODES)-LAP ;
%Initialize network
T_N_MINUS_1=T_START;
% Define DCO inputs at edge N
Y=zeros(N_NODES,1);
EPSILON_N=zeros(N_NODES,1);
% Define total error of each node at edge N-1
E_N_MINUS_1=zeros(N_NODES,1);
T_N=T_N_MINUS_1+T0;
T_OLD=T_N_MINUS_1(:,ones(1,N_NODES));
% Define error between all the nodes at edge N-1
ERRORS_OLD=T_OLD'-T_OLD;
DT=zeros(1,N_NODES);
N=1;
%Loop over edges
while N<N_EDGES,
        T_NEW=T_N (:,ones(1,N_NODES));
        % Define error between all the nodes at edge N
        ERRORS_NEW=T_NEW'-T_NEW;
        %Define M so that M(i,j)=1 if node i is leading
        %with respect to node j
        M=ERRORS_NEW>0;
        if strcmp(TYPE,’II’)
          E_N_MINUS_1=EPSILON_N;
        else
          E_N_MINUS_1=sum(A.*ERRORS_OLD,2);
        end
        EPSILON_N=sum(A.*(M.*ERRORS_OLD+(1-
        M).*ERRORS_NEW),2);
        % Update DCO input
        Y=Y+K1*EPSILON_N+K2*E_N_MINUS_1;
        DT=T0+Y;
        T_N=T_N+DT;
        ERRORS_OLD=ERRORS_NEW;
        N=N+1;
end
```