Research Repository UCD

Title	Semianalytical model for high speed analysis of all-digital PLL clock-generating networks
Authors(s)	Koskin, Eugene, Galayko, Dimitri, Feely, Orla, Blokhina, Elena
Publication date	2017-05-31
Publication information	Koskin, Eugene, Dimitri Galayko, Orla Feely, and Elena Blokhina. "Semianalytical Model for High Speed Analysis of All-Digital PLL Clock-Generating Networks." IEEE, May 31, 2017. https://doi.org/10.1109/ISCAS.2017.8050582.
Conference details	The 2017 IEEE International Symposium on Circuits and Systems (ISCAS), Baltimore, United States of America, 28-31 May 2017
Publisher	IEEE
Item record/more information	http://hdl.handle.net/10197/9824
Publisher's statement	© 2017 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.
Publisher's version (DOI)	10.1109/ISCAS.2017.8050582

Downloaded 2026-05-01 23:36:26

The UCD community has made this article openly available. Please share how this access benefits you. Your story matters! (@ucd_oa)

© Some rights reserved. For more information

Semianalytical Model for High Speed Analysis of All-Digital PLL Clock-Generating Networks

Eugene Koskin[1], Dimitri Galayko[2], Orla Feely[1] and Elena Blokhina[1]
[1]School of Electrical and Electronic Engineering, University College Dublin, Belfield, Dublin 4, Ireland
[2]LIP6 Lab, UPMC Sorbonne Universités, 4 place Jussieu, 75252 Paris Cedex 05, France

***Abstract*—In this paper, we propose the model of a network consisting of All-Digital Phase-Locked Loop Network in application to Clock-Generating Systems. The method is based on a solution of a system of non-linear finite-difference stochastic equations and allows us to perform high speed simulations of a distributed Clock Network on arbitrary topology. The result of our analysis show a good agreement with experimental measurements of a 65nm CMOS All-Digital Phase-Locked Loop Network.**

I. Introduction

A Phase-Lock Loop (PLL) Network is a spatially distributed array of PLLs interacting with each other by means of feedback error signals. It has been shown [1]–[6] that PLL networks can be used for generating a distributed clock signal in systems-on-a-chip (SoCs) where, due to increasing complexity, it becomes impossible to supply a subsystem by a coherent clock signal from a single crystal oscillator. The clocking signal is normally propagated within a microchip with the help of clock distribution trees. The primary disadvantages of this clock distributing approach is that numerous buffers in clock trees consume a lot of energy and have propagation delay due to relaxation processes in them. That results in the signal's distortion causing skew and jitter which is the major parasitic effects in clock generating networks.

A network can be built by allocating each oscillator in a node on a Cartesian grid. This represents a very common and most feasible case. Every node in a PLL network (fig. 1(a)) represents a single PLL that contains convention components such as: a voltage controlled oscillator (VCO) that usually generates a harmonic signal in a relatively wide range of frequencies; a control block – provides a control signal for VCO based on a phase-frequency error between VCO and its neighbours; a phase-frequency detector (PFD) that produces error signals for the corresponding control block. The number of PFDs for each oscillator is equal to the number of its neighbours. All these elements connected together result in a phase-frequency synchronicity of the network if the control parameters are properly chosen. It was shown [3] that, when at least one node in the network is connected to an external reference signal then the network might be synchronised with that signal both in frequency and phase. If the network is autonomous, i.e., does not have an external reference signal, then it synchronises to a common frequency that depends on its initial conditions.

When considering a networks of oscillators, one should differ the frequency synchronicity from the phase synchronicity. The frequency synchronicity depends on control parameters whilst the phase synchronicity depends also on a type of phase detector. It was shown in [1] that for analogue PLL networks such undesirable effects as mode-locking often take place. Therefore, it has to be exterminated by a proper choice of transfer characteristics of phase detectors that makes undesirable modes unstable.

It has been shown that using All-Digital PLLs (ADPLL) instead of analogue PLL in a network dramatically increases robustness of the synchronous regime of an ADPLL network with respect to mode-locking [5]. The major reason for that lies in a digital implementation of PFD. This detector compares time instances between positive edge trigger events of two input rectangular-wave signals rather than their phase difference. The detector's output represents a corresponding integer error value as a function of time difference between its input signals. For that reason we will call that particular type of PFD a digital time detector (DTD). Moreover, due to the fact that all components in ADPLLs are implemented digitally, the concept demonstrates a good scaleability with respect to manufacturing process.

Networks based on All-Digital PLLs are very complex, and a limited number of the general methods of their analysis have been developed yet. That is why the main tool for its studying and design is still behavioural modelling based on MATLAB Simulink or hardware/mixed-signal description languages. However, the simulations are time consuming and do not allow a full exploration of the design parameter space. In order to study the system faster, an analytic model has to be offered. But there are only a few analytical models that have been suggested so far to answer these questions, e.g., [5], [7], [8] and none of them are general enough to describe at least a single ADPLL in order to be implemented in a network.

The study [9] has developed a theoretical model for a single ADPLL that is general enough to capture important features of an ADPLL such as: self-sampling, RMS jitter, digital jitter, frequency and phase acquisition, phase tracking and dynamical instabilities. In this paper, we generalise our model of a single ADPLL for a case of ADPLL networks having arbitrary topology. Based on this model, we investigate the dynamics of the ADPLL network by varying the parameters of the system. The validation of the mathematical model is carried out through a series of experiments made on a chip that has been developed in [10] and shows a good agreement with the experimental results.

II. Statement of the Problem

An ADPLL network is an array of DCOs topologically connected with each other via DTD. The global reference signal is a part of the network and represented by a reference oscillator that affects other oscillators but can not be affected by any of them (see fig. 1(a), e.g., oscillators $1, 2$). The network can be described in terms of a directed graph $G = (V, E)$ with vertices V and edges E. All DCOs combined with their

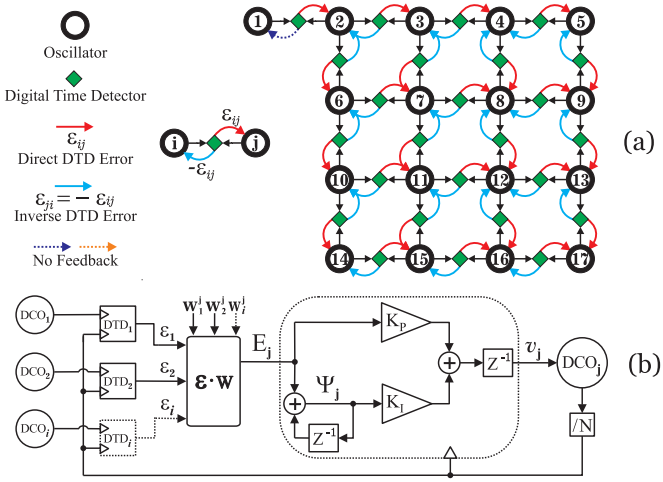

Fig. 1. Schematic view of an ADPLL network including its principal blocks: digital time detector (DTD), PI control block, digitally controlled oscillator (DCO) and reference signal

control blocks are allocated in vertices of a directed graph. The information about mutual pair-interconnection between oscillators is contained in a set of edges.

We assume that the global reference oscillator is allocated in the first vertice that has index $i = 1$. All the others DCOs are associated with vertices having indexes $i = 2 \ldots N + 1$, where N – number of a DCO in the network. Every $k-$edge in graph $k : \{i \to j\}$ connects two oscillator via DTD: $DCO[i] \to DTD[k] \leftarrow DCO[j]$. For every $DTD[k]$, the oscillator having the lower index $\min(i, j)$ we define as reference, the higher $\max(i, j)$ – local. Two outputs of DTD connected with control blocks of adjecent DCOs represent their timing errors. The direct error signal goes towards the local DCO and the inverse one – to the reference DCO. We can turn off a feedback control to the specified oscillator by setting a corresponding weight coefficient to zero.

In this study, we consider the ADPLL network based on a theoretical model of a single ADPLL developped previously in [9]. We employ the DTD as suggested in [5] as a combination of finite-state machine and time-to-digital converter. It detects a rising edge of the reference and the local oscillator, measures the time difference between them and computes an error in the form of digital signal in the range $[-7; -1] \cup [1; 7]$. The loop filter is realised as a proportional-integral (PI) controller and it provides a control signal for the DCO. The DCO generates a rectangular clock signal over a wide range of frequencies. The frequency changes linearly with respect to a control code having a constant frequency gain. The frequency lies within a range given by minimum and maximum values. Outside the range it saturates to the minimum/maximum frequencies.

III. The Model of an ADPLL Network

In this section, we introduce a novel model and its iterating algorithm for a discrete-time modeling of an ADPLL network on arbitrary topology. This model will allow us to study the synchronisation between the global reference oscillator and on-chip oscillating network.

A. A brief overview of a single ADPLL model

The model of a single ADPLL having DTD discussed previously has been introduced in [9]. In the current paper we will use interpretation where input signals of DTD represented by local and reference time instances L_n and R_n. L_n is the time difference between the next rising edge (event) of the local clock and current event. The current event can be either reference or local. That is why it is very important to emphasise that n is the instance number. It updates every time when the new rising edge comes to one of the corresponding inputs of a DTD. In the same manner we denote R_n as the time difference between the next rising edge of the reference clock and the current event.

Obviously, $L_n \in [0; T_n^L]$ and $R_n \in [0; T_n^R]$, where T_n^L and T_n^R are periods of local and reference signals. Due to a feedback loop that supplies the local signal to one of the inputs of DPD (see fig. 1(b)), T_n^L changes from one iterating step to another. This change occurs only if the step n corresponds to the local clock event. The detector's error $\varepsilon_n = \mathrm{H}(\tau_n, \tau_{TDC}, N_D)$ represents an integer number: $\varepsilon_n \in [-N_D; -1] \cup [1; N_D]$ that increases (decreases) by one if the operating time τ_n exceeds an integer number of the time-to-digital converter's resolution step τ_{DTC}. When $\tau_n \geq N_D \tau_{DTC}$ then the error saturates to its maximum value N_D and vice versa, when $\tau_n \leq -N_D \tau_{DTC}$ then $\varepsilon_n = -N_D$. The sign of τ_n can be negative when the local clock initiates the error measurement. The PI controller generates a control code to tune the frequency of DCO linearly within a given range of frequencies.

The system of equations that describes the behaviour of a single ADPLL in terms of L_n, R_n represents as follows:

$$\begin{cases} T_n^R = T^R \zeta_n(\sigma_R) \\ T_n^L = 1/[f_0 + \Delta f_{DCO}(K_p \varepsilon_n + K_i \psi_n)]\zeta_n(\sigma_L) \\ L_{n+1} = T_n^L \theta^-(L_n - R_n) + (L_n - R_n)\theta^+(L_n - R_n) \\ R_{n+1} = -(L_n - R_n)\theta^-(L_n - R_n) + T^R \theta^+(L_n - R_n) \\ m_{n+1} = \mathrm{sign}[m_n + \mathrm{sign}(L_n - R_n)] \\ \tau_{n+1} = \tau_n |m_n| + m_{n+1}[L_{n+1}\theta^+(L_{n+1} - R_{n+1}) + \\ \qquad R_{n+1}\theta^-(L_{n+1} - R_{n+1})] \\ \varepsilon_{n+1} = \mathrm{H}(\tau_n, \tau_{TDC}, N_D) \\ \psi_{n+1} = \psi_n + \varepsilon_n \theta^-(L_n - R_n) \end{cases} \quad (1)$$

where external noise is implemented by a product of the target period with a log-normal random variable $\zeta_n(\sigma) = \exp[N(0, \sigma)]$; $\theta^-(x)$ and $\theta^+(x)$ are Heaviside functions: $\theta^+(x) + \theta^-(x) = 1$, $\theta^+(x) - \theta^-(x) = \mathrm{sign}(x)$, Δf_{DCO} – gain DCO, K_p, K_i – proportional and integral gain factors, f_0 – initial frequency of DCO, ε_n – proportional error, ψ_n – cumulative error.

Algorithms in equation (1) will be used further as the main building block for the model of ADPLL network.

B. An extension of the single ADPLL model

In order to apply the model of a single ADPLL for a network, we have to determine the total number of DTDs and their connections with other oscillators.

For each pair of oscillators in the network we need to implement a DTD if at least one of the oscillator affects another one. For example, in the case of 2×2 network including a reference signal we can determine a graph $G = (V, E)$ such

as:

$$G(V,E) = \begin{cases} V = \{\,1,2,3,4,5\,\} \\ E = \{\,1 \to 2, 2 \to 3, 2 \to 4, \\ \qquad 3 \to 2, 3 \to 5, 4 \to 2, \\ \qquad 4 \to 5, 5 \to 3, 5 \to 4\,\} \end{cases} \tag{2}$$

It can be seen that this graph has nine directed edges: four of them are symmetrical ($i \leftrightarrow j$) and one more includes the influence of the global reference signal on a DCO ($1 \to 2$). That is why we need five DTDs to represent a network.

In order to generalise further ideas, we represent a network graph in terms of adjacency matrix $\mathbf{A}$:

$$A_{ij} = \begin{cases} 1 & \text{if } i \text{ is reference for } j \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

where $i, j = 1 \ldots N+1$ – oscillator number in the network. Adjacency matrix for the network (2) described above is:

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{pmatrix} \tag{4}$$

We can see that $A_{ij} = A_{ji}$ almost everywhere except of A_{01}. As we mentioned above, it reflects the fact that the global reference signal (V_0) affects the first DCO (V_1) but not vice versa. In ADPLL network, all elements that lie on the main diagonal of the matrix (A_{ii}) are equal to zero. Having the adjacency matrix determined, we can define an array of DTDs $\tilde{\mathbf{D}}$ which we will use further. The typical code for that looks as follows:

$$\begin{aligned} &\textbf{for } i = 1 \ldots N-1 \textbf{ do} \\ &\quad \textbf{for } j = i+1 \ldots N \textbf{ do} \\ &\qquad p = max(A_{ij},\, A_{ji}) \\ &\qquad \textbf{if } p \neq 0 \textbf{ then} \\ &\qquad\quad \tilde{D}_k = (i,\, j) \\ &\qquad\quad k \leftarrow k+1 \end{aligned} \tag{5}$$

Length of $\tilde{\mathbf{D}}$ represents a number of DTDs in the network D_N. Each $\tilde{D}_k$, $k = 1 \ldots D_N$ stores two values $D_k = (i,\, j)$, where $i < j$. According to our definition of reference and local oscillators in section II, the first element in D_k corresponds to the index of a reference oscillator, whilst the second one – to a local oscillator.

Having determined the topology of the network, we will assign the matrix of weight coefficients $\mathbf{W}$ in figure 1.(b). Here we assume that each DCO consumes an average error value from its neighbours. The total number of neighbours for each oscillator determines by adjacency matrix $\mathbf{A}$: $g_j = \sum_{i=1}^{N} A_{ij}$. Thus, the weight matrix determines as follows: if $g_j \neq 0 \Rightarrow W_{ij} = A_{ij}/g_j$, otherwise $W_{ij} = 0$. We have to mention that in a digital implementation it is costly to divide by 3. So, in the working prototype it was divided by 4 instead.

Having the weight matrix determined, the average error that supplied to a PI controller of each DCO in the network – E_j calculates as follows:

$$E_j = \sum_{i}^{N} \mathrm{sign}(j-i)\varepsilon_i W_{ij} \tag{6}$$

Once the way for calculating of average errors for each oscillator in the network has been determined, we can calculate a set of control codes in order to update corresponding DCOs frequencies. It can be done using the model developed earlier (1). However, instead of using ε_n and ψ_n in the second equation, we shall substitute them with average errors E_n, Ψ_n.

In the next section we will show the comparison of theoretical results with the experimental data.

IV. Experimental Validation and Discussion

The purpose of the experiment was to show feasibility of the algorithm proposed in section III for theoretical modelling and high-speed numerical analysis of ADPLL networks. It was done by comparing experimental measurements of the transient processes in ADPLL network with the theoretical results. The experiment was performed with the prototype of a 4×4 ADPLL network circuit having the described architecture designed and fabricated in 65nm CMOS technology. The architecture of the microchip has been described in [11] and [6]. The parameters of the implemented ADPLL network are given in table I.

TABLE I
Parameters of the ASIC implementation of the ADPLL

Experiment	Parameter
VDD	1.0 V
Feedback division coef. N	2
Δf_{DCO}	2×150 kHz
$F_{min\ DCO}$	2×135 MHz
$F_{max\ DCO}$	2×175 MHz
τ_{TDC}	20 ps*
N_D	7

* Obtained by simulation

In order to observe a transient process on the oscilloscope, the input (reference) signal frequency was modulated by a rectangular wave. In this way, the reference signal frequency switched between minimum and maximum values (see fig. 2(a)). After each switching of the frequency value, the reference frequency remained constant for enough time to observe the frequency acquisition of the divided signal. The transient process is finished before the next switching, so that such a configuration generated a repeated sequence of identical transient process waveforms (see an example of the observed plots in fig. 2(a)). Such an experiment was repeated for different values of the filter coefficient K_i and K_p and network topology.

We carried out two series of experiments for a transient process where the frequency acquisition and the frequency synchronisation take place. The first series of experiments were made on the 2×2 ADPLL network and the results can be seen in fig. 2(b)-(d). We implemented this network by switching off all interactions between DCOs except those of $2, 3, 6, 7$ in fig. 1(a), where the global reference signal is represented by the first oscillator. The experimental data has been recorded from oscillators $2, 3$ and 7. After that,

the MATLAB script transformed them into time-frequency dependence.

In these experiments we observed both the frequency acquisition and synchronisation with the global reference signal. Moreover, it can be seen that the frequency synchronization between all DCOs in the network has fast dynamics. Due to the fact that all initial frequencies of DCOs (but not phases!) are the same, the synchronisation between them happens almost instantly just after the device switches on. Following that, the whole network behaves as if it is the one single oscillator.

In the second series of experiments we observed the frequency dynamics for the 4×4 ADPLL network. Due to restriction in the frequency modulation of the source generator that produces the global reference signal, only one experiment has been chosen to show frequency dynamics. The results of the experiment can be seen in the fig. 2(e). Ceteris paribus, the frequency acquisition rate in 4×4 network is almost four times slower than in 2×2 network. In order to capture it we had to boost the frequency acquisition rate by increasing an integral gain factor K_i. However, here we faced two kinds of restrictions. On the one hand, having a constant proportional gain factor K_p and increasing K_i we forced the network to leave a stable operating region in the parameters domain (K_p, K_i). On the other hand, if we increased K_p in order to return the system back to a stable region, the digital jitter increased at the same time. That is why we chose control parameters shown in fig. 2(e).

As it was shown in [9], the discrete-time models have a big advantage with respect to calculating performance. The model we presented here calculates $1ms$ of 4×4 ADPLL network only for $1s$ of real time on a typical PC (processor: 2.6 GHz Intel Core i5, memory: 16 GB 1600 MHz DDR3), which is $\approx 10^4$ times faster than using MATLAB Simulink framework. That reveals new opportunities for studying ADPLL networks. For example, the analysis of optimal control parameters providing a trade-off between low jitter and fast frequency acquisition rate in ADPLL networks, or the design of the optimal network topology that produces a stable clocking signal and minimises power consumption.

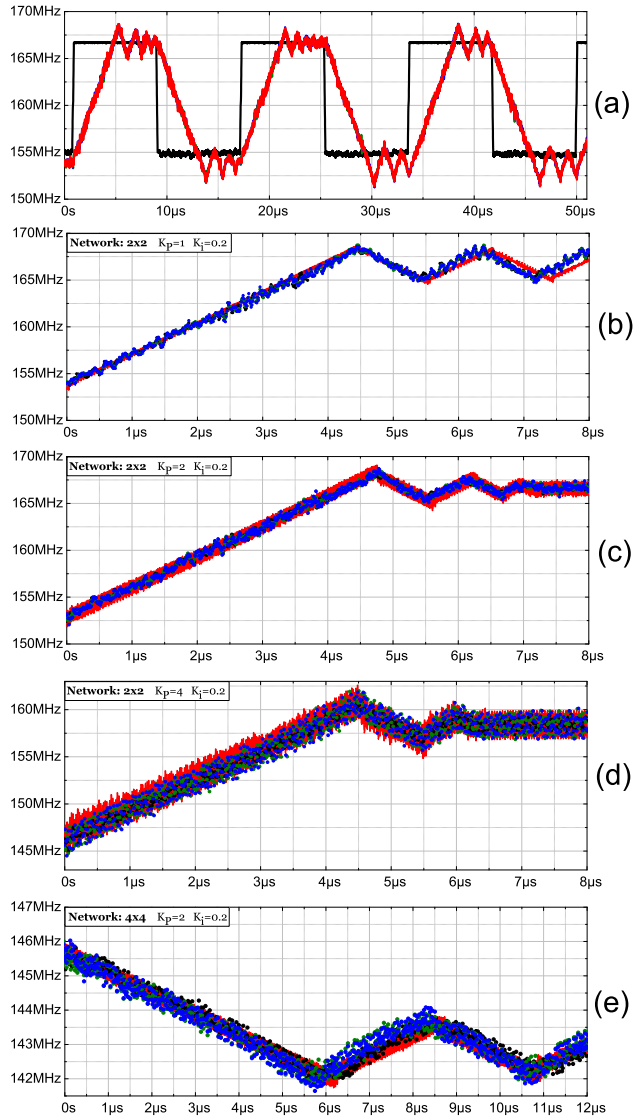

Fig. 2. Modulated input (black) and the ADPLL transient (red) in the experiment (a); the comparison of the experimental results (black, green, blue dots) and the discrete-time model (red lines) for 2×2 ADPLL network (b–d); the same for 4×4 ADPLL network (e);

Acknowledgment

The authors gratefully acknowledge the support of Science Foundation of Ireland (SFI).

References

[1] G. Pratt, J. Nguyen *et al.*, "Distributed synchronous clocking," *EEE Transactions on Parallel and Distributed Systems*, vol. 6, no. 3, pp. 314–328, 1995.

[2] E. G. Friedman, "Clock distribution networks in synchronous digital integrated circuits," *Proceedings of the IEEE*, vol. 89, no. 5, pp. 665–692, 2001.

[3] V. Gutnik and A. Chandrakasan, "Active GHz clock network using distributed PLLs," in *Proc. of International Conference on Solid-State Circuits (ISSCC)*, 2000, pp. 174–175.

[4] J. Akré, J. Juillard, M. Javidan, E. Zianbetov, D. Galayko, A. Korniienko, and E. Colinet, "A design approach for networks of self-sampled All-Digital Phase-Locked Loops," in *Circuit Theory and Design (ECCTD), 2011 20th European Conference on*, Aug 2011, pp. 725–728.

[5] J.-M. Akre, J. Juillard, D. Galayko, and E. Colinet, "Synchronization analysis of networks of self-sampled All-Digital Phase-Locked Loops," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 59, no. 4, pp. 708–720, 2012.

[6] C. Shan, D. Galayko, F. Anceau, and E. Zianbetov, "A reconfigurable distributed architecture for clock generation in large many-core SoC," in *Reconfigurable and Communication-Centric Systems-on-Chip (ReCoSoC), 2014 9th International Symposium on*, 2014, pp. 1–8.

[7] N. Da Dalt, "A design-oriented study of the nonlinear dynamics of digital bang-bang PLLs," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 52, no. 1, pp. 21–31, 2005.

[8] C.-Y. Zhang, Y.-b. Li, H.-M. Li, Q. Wang, and Y.-l. Peng, "The implementation and analysis of a new self-sampling pi control All Digital Phase-Locked Loop," in *International Conference on Machine Learning and Cybernetics, 2006*, 2006, pp. 241–246.

[9] E. Koskin, E. Blokhina, C. Shan, E. Zianbetov, O. Feely, and D. Galayko, "Discrete-time modelling and experimental validation of an all-digital pll for clock-generating networks," in *2016 14th IEEE International New Circuits and Systems Conference (NEWCAS)*, June 2016, pp. 1–4.

[10] C. Shan, E. Zianbetov, F. Anceau, O. Billoint, and D. Galayko, "A distributed synchronization of all-digital PLLs network for clock generation in synchronous SOCs," in *New Circuits and Systems Conference (NEWCAS), 2015 IEEE 13th International*, June 2015, pp. 1–4.

[11] E. Zianbetov, D. Galayko, F. Anceau, M. Javidan, C. Shan, O. Billoint, A. Korniienko, E. Colinet, G. Scorletti, J. Akré, and J. Juillard, "Distributed clock generator for synchronous SoC using ADPLL network," in *Custom Integrated Circuits Conference (CICC), 2013 IEEE*, Sept 2013, pp. 1–4.