Discrete-Time Modelling and Experimental Validation of an All-Digital PLL for Clock-Generating Networks

Eugene Koskin[1], Elena Blokhina[1], Chuan Shan[2], Eldar Zianbetov[3], Orla Feely[1] and Dimitri Galayko[2]
[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
[3] CEA, INAC-SPINTEC, F-38000 Grenoble, France

***Abstract*—In this paper, we derive a mathematical model of an All-Digital Phase-Locked Loop (ADPLL) employing a time-to-digital phase detector. The model we suggest represents a nonlinear discrete-time map and provides significant benefits for the simulation of a single PLL, a network of PLLs or their design. In particular, the model allows us to take into account the jitter of the reference and local clocks and other noises. The mathematical model (the map) is then compared with a behavioural model implemented in MATLAB Simulink and displays identical results. The simulation of the mathematical and behavioural models are further compared with experimental measurements of a 65nm CMOS ADPLL and show a good agreement.**

I. Introduction

All-Digital Phase-Lock Loops are widely used for frequency synthesis in modern low-power electronics [1], [2]. They can be found in microprocessors, radio receivers, mobile telephones, GPS systems, etc. In the last two decades, the application of PLLs has expanded beyond the use of a single PLL and now includes PLL based networks. These networks serve for generating a *distributed clock signal* in microprocessors or in systems-on-a-chip (SoCs) where, due to complexity, it becomes impossible to supply a subsystem by a clock signal from a single crystal oscillator. Starting from [3], researchers have been exploring the idea of PLL networks as a clock generating system [4]. In particular, ADPLL based networks are suggested in [5].

A PLL/ADPLL network is a complex system that exhibits synchronisation, mode locking and emerging behaviour. The main tool for the analysis and design of PLL networks is still behavioural modelling based on MATLAB Simulink or hardware/mixed-signal description languages. While the questions related to the stability of a PLL node (or a network as a whole) and its synchronisation error are of major importance, there are only a few analytical methods that have been suggested so far to answer these questions [5]–[7]. This is because certain features of ADPLLs are very difficult to incorporate into mathematical models. For instance, *self-sampling* is one of the most challenging nonlinear features of mixed analog/digital systems, and, to our knowledge, no standard general tool for its analysis exists. Self-sampling is a property that makes the system time variant and a few studies have addressed it. Without a correct and reliable theoretical model of an ADPLL, it would be impossible to build on, and study a network. In addition, the simulation time of a behavioural model is significantly longer than that of a discrete-time iterative model and does not allow a full exploration of the design parameter space.

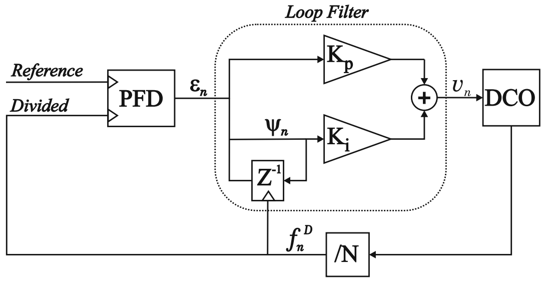

Fig. 1. Schematic view of an ADPLL including its principal blocks: phase-frequency detector (PFD), loop filter (proportional-integral), digitally controlled oscillator (DCO) and frequency divider (/N).

In a common theoretical ADPLL model (presented, for instance, in Ref. [6]), the detector's error is represented as a signum function of the time intervals between *signals* — rising edges of the reference and divided clock. Analytically, it is represented as: $\varepsilon_k = \mathrm{sgn}(\Delta t_k)$, $\Delta t_k = t_{r,k} - t_{d,k}$, where k is the sampling instant that corresponds to the k^{th} divided clock signal, $t_{d,k}$ is the time of the k^{th} divided signal (D), and $t_{r,k}$ is the time of the k^{th} reference signal (R). By definition, each sampling instant k generates a basic pair of signals: either divided signal is followed by the reference signal or vice versa (RD or DR). These pairs of signals are merged together for $k = 0, 1, 2\ldots$ and form the general sequence of the process. As a result of this, any triple or higher sequence of the same signal such as RRR or DDD can not be expressed within this model. The common model works well for a PLL in a synchronised mode, but any clock sequence that violates that rule makes the model inconsistent. In reality, such sequences can occur in a variety of cases. For example, during the frequency acquisition phase, the frequency of the divided clock signal is far from the reference signal and we will observe many sub-sequences of the following type: RRR or DDD. Similar sequences can appear when the system losses stability.

In this paper, we present a novel approach for modeling such a system using a combined signal sequence, which serves as the basis for our discrete-time model. The model we suggest is general enough to capture RMS jitter as well as digital jitter, frequency acquisition and dynamical instabilities. Based on the model, we investigate the dynamics of the ADPLL by varying the parameters of the system. The validation of the mathematical model is carried out through a behavioural model implemented in MATLAB Simulink and through a series of experiments. The discrete-time model we derive shows an absolute agreement with our behavioural model and excellent agreement with the experimental results.

II. Statement of the Problem

An ADPLL (fig. 1) consists of a phase-frequency detector (PFD), a loop filter (LF), a digitally controlled oscillator (DCO) and a frequency divider (FD). All elements of the system are implemented digitally. The role of the PFD is to compare the phases between two input signals: the reference signal (external) and the divided signal (from the DCO). It then generates an error signal as a function of the phase (time) difference. This error signal passes through the LF and controls the frequency of the DCO. Finally, the signal from the DCO is supplied back to the PFD through the FD and a feedback loop minimises the phase error. This leads to both frequency and phase synchronicity between the reference and divided signals, whereas the DCO's frequency appears to be multiplied by the divider's factor. This multiplication principle allows one to generate a stable frequency required for a specific application. For instance, the reference signal may come from a piezoelectric oscillator that provides a relatively low but stable frequency. A free running DCO generates a much higher but unstable frequency. However, embedded into a PLL, it provides a stable signal with a small frequency variance.

In this study, we consider an ADPLL model that employs a digital phase detector as suggested in [5]. It detects the rising edge of the reference and the local (DCO) signals, measures the time difference between them and computes an error signal in the form of a 3-bit digital signal. In order to exclude incorrect measurements (for instance, the time between two rising edges of the reference signal only), it is implemented as a finite-state machine. The loop filter is realised as a proportional-integral (PI) filter and it provides a control code for the DCO. The DCO generates a rectangular clock signal over a wide range of frequencies. The frequency changes linearly with a constant frequency step from a minimum to a maximum value with respect to a control code.

III. Mathematical Discrete-Time Model

In this section, we suggest a new mathematical discrete-time model of the ADPLL. In addition, this ADPLL has been implemented in MATLAB Simulink and compared with our mathematical model and experimental results. Here we study the synchronisation between the reference and divided signals. In this case, the local clock signal represents the time scaled divided signal with the scaling factor $1/N$.

A. Clock Signals Representation

We denote every rising edge of the reference clock (R) and the divided clock (D) as clock signals. The period of each clock is supposed to be constant. The sequence of the signals (reference and the divided) are then merged into one sequence,

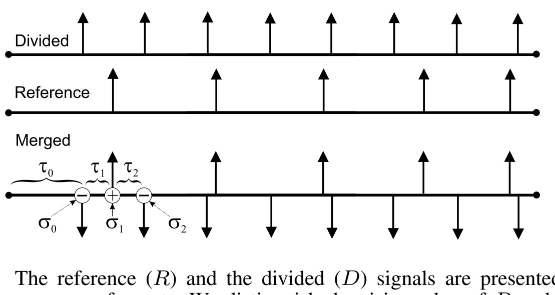

Fig. 2. The reference (R) and the divided (D) signals are presented as a merged sequence of events. We distinguish the rising edge of R and D by assigning the signature σ, where '+1' corresponds to R and '-1' corresponds to D.

Σ where we distinguish R and D by assigning a signature σ: '$+1$' corresponds to R and '-1' corresponds to D. We denote the time interval between two consecutive signals as τ. By taking this into account we represent a sequence Σ as ordered pairs of (τ_n, σ_n) (fig. 2). Further, we will show that Σ can be represented as the result of a specific recursive procedure.

Let us consider t_n, the time of a signal (either R or D), where n is the signal's number in the sequence. Obviously, the time to the next *closest* signal regardless of its type is $\tau_n = \min(D_n, R_n)$, and its signature is $\sigma_n = \operatorname{sgn}(D_n - R_n)$. Assuming that the next closest signal is the divided signal ($\sigma_n = -1$), the following divided clock signal (D_{n+1}) comes after one divided clock period: $D_{n+1} = T^D$, whereas the time to the nearest reference clock signal decreases by D_n: $R_{n+1} = R_n - D_n$. Comparatively, when ($\sigma_n = 1$) we get: $D_{n+1} = D_n - R_n$, $R_{n+1} = T^R$.

We can generalise these conclusions in the form of a map:

$$\begin{cases} D_{n+1} = T^D\theta^-(D_n - R_n) + (D_n - R_n)\theta^+(D_n - R_n) \\ R_{n+1} = -(D_n - R_n)\theta^-(D_n - R_n) + T^R\theta^+(D_n - R_n) \end{cases} \tag{1}$$

where $\theta^+(x)$ is the Heaviside step function and $\theta^-(x) = \theta^+(-x)$. We can simplify (1) by introducing the variable transforms $2\eta_n = D_n - R_n$ and $2\xi_n = D_n + R_n$:

$$\begin{cases} \eta_{n+1} = \left(\frac{T^D}{2} + \eta_n\right)\theta^-(\eta_n) + \left(\eta_n - \frac{T^R}{2}\right)\theta^+(\eta_n) \\ \xi_{n+1} = \left(\frac{T^D}{2} - \eta_n\right)\theta^-(\eta_n) + \left(\eta_n + \frac{T^R}{2}\right)\theta^+(\eta_n) \end{cases} \tag{2}$$

From the new variables ξ_n and η_n, the time τ_n to the next closest signal and its signature σ_n transform as follows:

$$\begin{aligned} \tau_n &= \xi_n - |\eta_n| \\ \sigma_n &= \operatorname{sign}(\eta_n) \end{aligned} \tag{3}$$

Equations (2) and (3) describe a free running system (no feedback) with the reference and divided signals mapped to a common time scale. It should be mentioned that it is impossible to represent the process as a recursion between $(\tau_n, \sigma_n) \mapsto (\tau_{n+1}, \sigma_{n+1})$ due to the non bijective map $(\eta_n, \xi_n) \to (\tau_n, \sigma_n)$. That is why we shall refer to η_n and ξ_n as primary state variables.

Our next step is to describe the PFD, the first block towards a feedback control of the DCO.

B. Digital Phase-Frequency Detector

The digital PFD is implemented as a finite-state machine whose diagram is shown in fig. 3(a). Its state is described by two variables: the state of the detector – s and its mode – $\hat{m}$. Variable s reflects the leading clock. If the divided signal leads, then $s = 0$, whereas if the reference clock leads, then $s = 1$. The variable $\hat{m}$ represents the operational mode of the

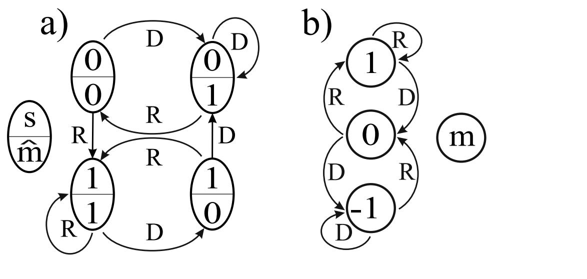

Fig. 3. Phase-frequency detector from [5] as a finite-state machine (a) and its equivalent representation (b).

PFD. When it is in the measurement mode, $\hat{m} = 1$, whereas when it is in the waiting mode, $\hat{m} = 0$. It can be shown, that the PFD of fig. 3(a) can be equivalently represented as in fig. 3(b) with the state variable m that can take three values: $m = \{-1, 0, 1\}$. There is a correspondence between the original description of the PFD in terms of $(s, \hat{m})$ and the modified description in terms of (m): $(0,0) \to 0$, $(0,1) \to -1$ and $(1,1) \to 1$. The advantage of this transform is that we now describe the PFD using only one variable and use the following analytical description:

$$m_{n+1} = \frac{m_n}{2} + \sigma_n \left(1 - \frac{m_n^2}{2}\right) \tag{4}$$

where m_n denotes the PFD state just before the n^{th} signal. The detector does not change its state in case of two or more signals of the same type (with the same σ) occurring in the merged sequence Σ. Thus, the merged sequence Σ is very convenient here as it allows us to reproduce the memory-dependent behaviour of the PFD in a simple manner.

C. Proportional and Integral Errors ε and ψ from the PFD

The digital PFD described in [5] provides an integer value of the timing error $\varepsilon_n = \pm 1, \pm 2, \ldots, \pm N_D$. The timing error ε_n has the sign '$+$' when a reference signal triggers the measurement mode and the sign '$-$' when a divided signal does so. N_D is the maximum error that the PFD can provide. The timing error is a function $\varepsilon = \mathrm{H}(\tau^{op}, \tau_{TDC}, N_D)$ with

$$\mathrm{H} = \mathrm{sgn}(\tau^{op}) \min(\lceil \tau^{op}/\tau_{TDC} \rceil, N_D) \tag{5}$$

where τ_n^{op} is the operating time of the PFD (the time between two instants when m changes from $0 \to 1$ and from $1 \to 0$) and τ_{TDC} is the time resolution step of the PFD. The sign of τ_n^{op} depends on the triggering signal.

Now we define the process that accumulates the operating time τ_n^{op} of the PFD in our model. If it is in the measurement mode and the same type of signals occur in the sequence Σ, the state of the PFD does not change. Thus, the operation time increases. If the PFD is in the waiting mode, it holds the last measured value, and any signal occurring next triggers the measurement mode. This process is shown in fig. 4. Therefore,

$$\tau_{n+1}^{op} = m_n^2 \tau_n^{op} + \tau_{n+1} m_{n+1} \tag{6}$$

The detector provides an output timing error which is ready fore use just after the next signal:

$$\varepsilon_{n+1} = \mathrm{H}(\tau_n^{op}, \tau_{TDC}, N_D) \tag{7}$$

The accumulation of the errors in the integral path of the PI filter occurs only at the divided clock signal. This gives the equation for the integral error ψ_n:

$$\psi_{n+1} = \psi_n + \theta^-(\eta_n)\varepsilon_n \tag{8}$$

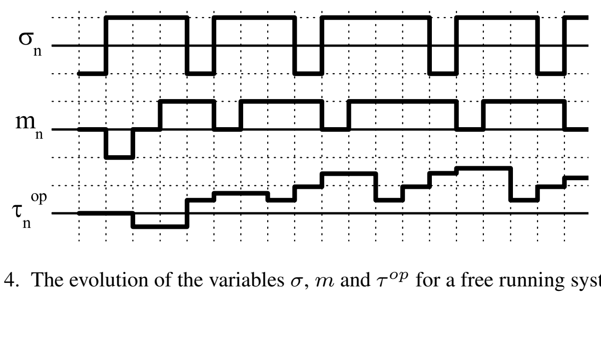

Fig. 4. The evolution of the variables σ, m and τ^{op} for a free running system.

D. The Frequency Control Procedure

Due to the self-sampled nature of the ADPLL, the frequency of the divided signal f^D changes only at the divided clock signal, when $\sigma_n = -1$. The DCO suggested in [5] is designed to generate a range of frequencies with a constant frequency step, f_{DCO}. The number of steps determines the DCO bit resolution. Beyond this range, frequency saturation takes place. However, it is neglected here because all the experiments lie within the control range. Therefore, the divided frequency can be represented as: $f_n^D = f_0 + \Delta f_{DCO} v_n$, where f_0 is the initial divided frequency of the DCO, Δf_{DCO} is the divided gain of the DCO and v_n is the control signal we obtain after the PI filter. The signal v_n represents a linear combination of the proportional (timing) and integral (cumulative) errors ε_n and ψ_n: $v_n = K_p \varepsilon_n + K_i \psi_n$ where K_p and K_i are the gains of the proportional and integral paths of the PI filter.

It is straightforward to generalise the model with respect to RMS jitter. For this, we add a random fluctuation to the signal frequency just after the corresponding signal occurred: $f_n^R = f^R \exp[\mathrm{N}(0, \sigma_R)\theta^+(\eta_n)]$, $f_n^D = f^D \exp[\mathrm{N}(0, \sigma_D)\theta^-(\eta_n)]$, where f^R and f^D are noiseless frequencies, $\mathrm{N}(0, \sigma)$ is a random number obtained from a Gaussian distribution with zero mean and standard deviation σ. The quantity σ_D corresponds to the divided signal and σ_R – to the reference signal.

E. Final Equations Taking into Account Jitter

By collecting all the equations that describe each block of the PFD, we manage to involve a feedback loop and obtain the *discrete-time* model (map) of the ADPLL:

$$\begin{cases} f_n^R = f^R \exp[\mathrm{N}(0, \sigma_R)\theta^+(\eta_n)] \\ f_n^D = [f_0 + \Delta f_{DCO}(K_p \varepsilon_n + K_i \psi_n)] \exp[\mathrm{N}(0, \sigma_D)\theta^-(\eta_n)] \\ \eta_{n+1} = \left(\frac{1}{2f_n^D} + \eta_n\right)\theta^-(\eta_n) + \left(\eta_n - \frac{1}{2f_n^R}\right)\theta^+(\eta_n) \\ \xi_{n+1} = \left(\frac{1}{2f_n^D} - \eta_n\right)\theta^-(\eta_n) + \left(\eta_n + \frac{1}{2f_n^R}\right)\theta^+(\eta_n) \\ m_{n+1} = \frac{m_n}{2} + \mathrm{sign}(\eta_n)\left(1 - \frac{m_n^2}{2}\right) \\ \tau_{n+1}^{op} = m_n^2 \tau_n^{op} + m_{n+1}(\xi_{n+1} - |\eta_{n+1}|) \\ \varepsilon_{n+1} = \mathrm{H}(\tau_n^{op}, \tau_{TDC}, N_D) \\ \psi_{n+1} = \psi_n + \theta^-(\eta_n)\varepsilon_n \end{cases} \tag{9}$$

This discrete-time model describes the state of the ADPLL from one clock signal in the merged sequence Σ to another one, thus representing the time evolution of the ADPLL. Note that we require initial conditions to start the iterative process (9). Usually, they are: $\tau_0 = 0$, $\varepsilon_0 = 0$, $\psi_0 = 0$, $m_0 = 0$, with η_0 and ξ_0 depending on the initial phases of the divided and reference signals.

IV. Results and Discussion

A prototype of a single ADPLL circuit having the described architecture has been designed and fabricated in 65nm CMOS technology and shown in figure 5. The architecture has been specifically intended as a building block of scalable networks for ADPLLs as described in [8] and [9]. The parameters of the implemented ADPLL are given in table I. The goal of the experiment is to highlight the relationship between the nature of the transient process and the parameters of the PI filter. To achieve, the ADPLL is implemented with the programmable coefficients K_i and K_p. The measurements have been taken from a single ADPLL connected to a reference clock while leaving all the others inactive.

TABLE I
PARAMETERS OF THE ASIC IMPLEMENTATION OF THE ADPLL

Parameter	Value
VDD	1.0 V
Feedback division coef. N	4
Δf_{DCO}	4×156 kHz
$F_{min\ DCO}$	4×135 MHz
$F_{max\ DCO}$	4×175 MHz
τ_{TDC}	20 ps*

* 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 the values 143 MHz and 167 MHz every 7.5 μs. After each switching of the frequency value, the 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 6(a)). Such an experiment is repeated for different values of the filter coefficient K_i and K_p. Figure 6(b) and fig. 6(c) show a zoomed segment of the transient process obtained in the experiment for two sets of the filter parameters (black dots). The result is then compared with the proposed analytical model parameterised by the same filter coefficient values used in the experiment (red dots). Moreover, the Simulink results perfectly coincidence with the analytical results in a jitterless case (blue line). This means that the algorithms in the Simulink model were mapped correctly.

The major advantage of the discrete-time model is that the simulations are very fast. For instance, in order to do a simulation in MATLAB Simulink, an appropriate calculation's time step has to be chosen. Normally, it is less than the smallest time scale parameter of the system. In our case it is determined by $\tau_{TDC} = 20\ ps$. The simulation time for a set of experiments is 5 μs. Therefore, in order to get the results in Simulink, $\sim 10^6$ simulation steps are needed, whereas the suggested map model needs only $\sim 10^3$ iterating steps. More precisely, in order to simulate 5 μs for the map (9), 1600 iterations needed. On a typical PC (processor: 2.6 GHz Intel Core i5, memory: 16 GB 1600 MHz DDR3) $3 \cdot 10^{-4}$ s are needed. While for the Simulink model (solver: ode45, max step size: 10^{-11} s), it takes about 5 s of computational time which is $\sim 10^4$ times slower.

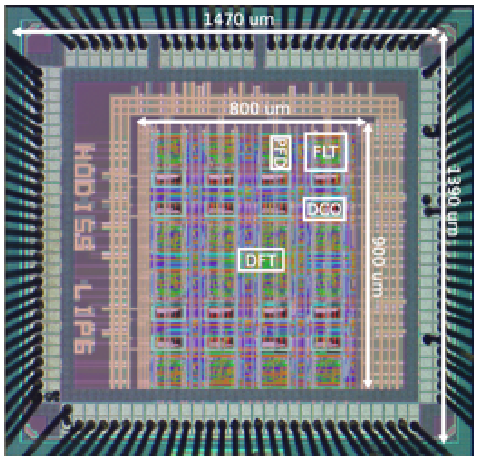

Fig. 5. Die microphotograph of 4×4 coupled oscillators ADPLLs network. The proof-of-concept chip generating 1.1 – 2.4 GHz clock is implemented in 65 nm CMOS technology developed by Dr Galayko's group.

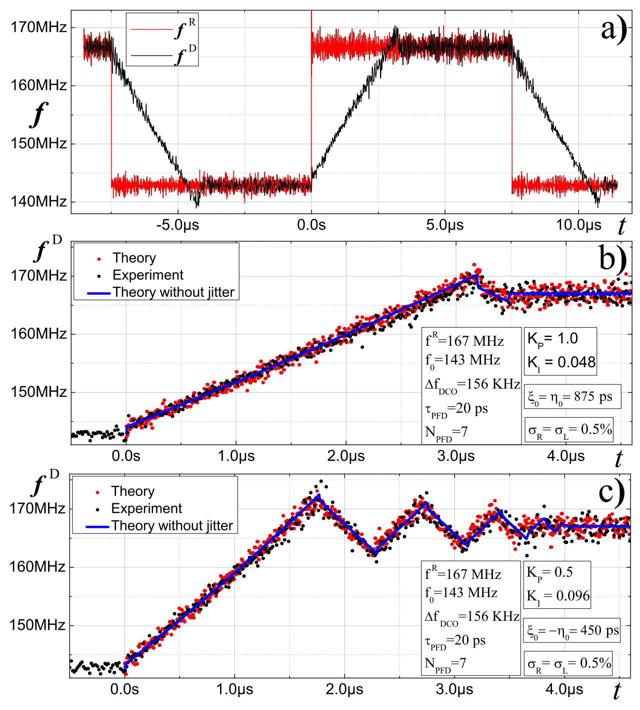

Fig. 6. (a) Modulated input and the ADPLL transient in the experiment; (b) and (c) is the comparison of the experimental results (black dots) and the discrete-time map (9) (red dots). The blue line corresponds to a jitterless case of both the MATLAB Simulink behavioural model and the discrete-time map.

Finally, the discrete-time map allows an analytical investigation of a single ADPLL as well as a fast simulation of an ADPLL network including stochastic effects.

ACKNOWLEDGMENT

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

REFERENCES

[1] T. Olsson and P. Nilsson, "A digitally controlled pll for SoC applications," *IEEE Journal of Solid-State Circuits*, vol. 39, no. 5, pp. 751–760, 2004.

[2] R. B. Staszewski and P. T. Balsara, *All-Digital Frequency Synthesizer in Deep-Submicron CMOS*. Wiley-Interscience, 2006.

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

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

[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] 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.

[7] 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.

[8] 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.

[9] 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.