FPGA prototyping of large reconfigurable ADPLL network for distributed clock generation

Chuan Shan, Eldar Zianbetov, Weiqiang Yu, François Anceau, Olivier Billoint, Dimitri Galayko

HAL Id: hal-01053762

https://hal.sorbonne-universite.fr/hal-01053762v1

Submitted on 4 Aug 2014

FPGA Prototyping of Large Reconfigurable ADPLL Network for Distributed Clock Generation

Chuan Shan [1], Eldar Zianbetov [1], Weiqiang Yu [1], François Anceau [1], Olivier Billoint [2], Dimitri Galayko [1]

[1] Laboratoire d'informatique de Paris 6 (LIP6)
University of Pierre and Marie Curie (UPMC), Paris, France
[2] CEA-LETI, Grenoble, France
Email: [1]{First name.Last name}@lip6.fr, [2]{First name.Last name}cea.fr   Tel: +33 (0)1 44 27 70 16

Abstract—In this paper, we present an FPGA platform for the design and study of a network of coupled all-digital phase locked loops (ADPLLs). An implementation of a programmable and reconfigurable 10×10 ADPLL network is described, which is designed for prototyping distributed clock generation in large synchronous system on chip (SoC). The paper emphasizes the reconfigurability of proposed system, which allows exploiting stability issues and nonlinear behavior of a $N \times M$ network of coupled oscillators (the dimension can be configured from 1×1 to 10×10). Configurations with different parameters are compared and analyzed. A dynamic setup mechanism is proposed, allowing selecting the desired synchronized state. Experimental results validate theoretical analysis about circuit parameters and demonstrate the global synchronization of network and performance for different configurations.

Keywords-Synchronization, ADPLL, FPGA, prototyping, reconfigurability.

I. INTRODUCTION

Clocking is essential for large system on chip (SoC) like multi-core processors. The main clock related issue is the difficulty of implementing a global centralized clock distribution in advanced deep submicron CMOS technologies. For this reason, nowadays large SoCs are partitioned into several small synchronous clocking areas (clock domains), and the communication between the logic blocks in different clock domains is asynchronous (Globally Asynchronous Locally Synchronous architecture). This solution presents several drawbacks, particularly in what concerns chip design, verification and by consequence, system reliability. A fully-synchronous design requires a synchronization between the local clocks. For this, instead of using a centralized clock distribution, our study explores an alternative technique of distributed clock generation: a network of coupled and synchronized oscillators for the generation of local clocks of the synchronous clocking areas (SCA). If the clock generators of each clock domain are synchronized in phase and in frequency, synchronous communications are possible between the zones. In our study, the synchronization between the oscillators is achieved through the network of coupled all-digital phase locked loops (ADPLL). The advantage of this solution is a use of only local links between the oscillators, and exploitation of the smartness of digital circuits for management of oscillator synchronization, comparing with previous analog implementations [1].

The main challenge in implementation of this solution is the design of a large size coupled ADPLL network: this is a multidimensional nonlinear digital system, whose stability and performance must be controlled by digital correctors embedded in the network nodes. This nonlinear system has many degrees of freedom, and by consequence, it has multiple steady operation modes depending on system coefficients and initial conditions. To study these issues, we need a prototype to model as precisely as possible the behavior of an ASIC-based ADPLL network. Because of digital implementation of the system, the FPGA prototyping appears to be an appropriate solution.

This work presents an FPGA implementation of 10×10 network of coupled oscillators, which is a prototype for a chip in 65nm CMOS technology whose design is ongoing. The FPGA model of the ADPLL network has the same architecture and the same parameter values as the future CMOS ASIC circuit, however, scaled down proportionally in frequency because of the maximum frequency limit of the FPGA platform. The analog/digital blocks of the ADPLL (the digitally controlled oscillator and the phase-frequency detector) have a particular implementation adapted to the FPGA limitations.

In section II the architecture of the 10×10 network and its functional blocks are described. The procedure allowing homothetic sizing of FPGA prototype with respect to the ASIC prototype is also presented in this section. Section III discusses about how we use the system reconfigurability feature to verify the relation between different parameters and system performance, and how to avoid undesired steady states. Section IV presents experimental results to validate our theoretical analysis.

II. NETWORK ARCHITECTURE

The topology of the 10×10 network is presented in Fig. 1. It is composed of 100 filter/oscillator (FO) blocks and one phase frequency detector (PFD) between each two neighboring synchronous clock areas (SCA) for measuring the phase error between each coupled oscillators (180 PFDs). The PFD placed in the upper-left corner compares the phase of the oscillator clock in SCA1-1 with the input reference clock. Such a network, if properly designed, is synchronized: it means that the all oscillators have the same frequency and phase as
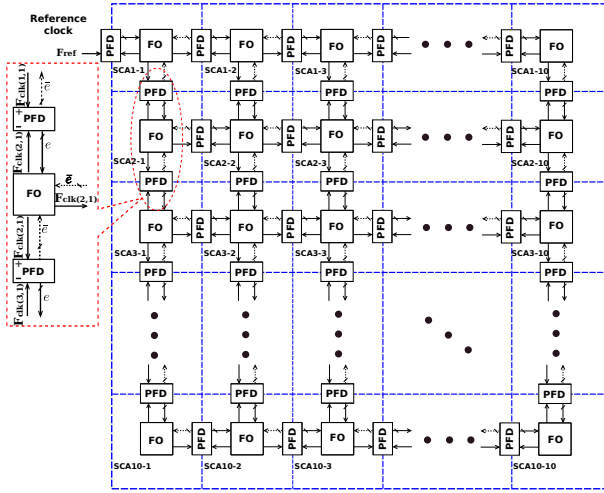
Fig. 1: 10×10 network of coupled PLL

the input reference signal, and residual synchronization phase errors are small and limited in a well-defined range.

The structure of a network node is presented in Fig. 2 [2]. Each node contains 2, 3 or 4 PFDs: each PFD detects the phase/frequency difference between the locally-generated clock and a neighboring clock. The number of PFDs depends on the position of node. A PFD generates a 5-bit signed binary code. For each node, the errors with neighbors are added and processed by the loop filter (LF), so as to generate a 10-bit control word for the digitally controlled oscillator (DCO). The loop filter is a Proportional-Integral (PI) filter.
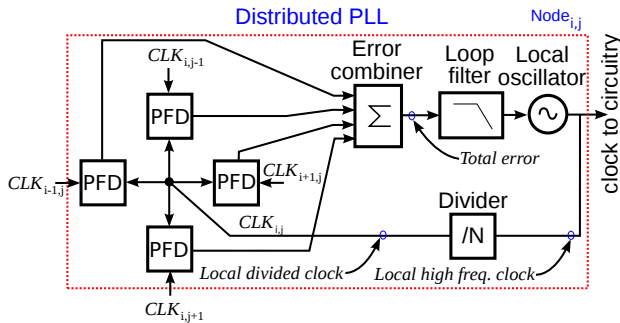

Fig. 2: Architecture of ADPLL

A. PFD

Fig. 3 presents the structure of the PFD. The Bang-Bang Phase-Frequency Detector (BB-PFD) detects the sign of the phase error ($SIGN$) and the interval of it ($MODE$) [3]. The duration of the signal $MODE$ represents the absolute value of the phase error. It is applied to the input of TDC, which converts the duration of input signal to an unsigned binary code ($Dout$), which is then combined with the $SIGN$ signal by the arithmetic block to form a signed binary code ($ERROR$). The transfer function is presented in Fig. 4.
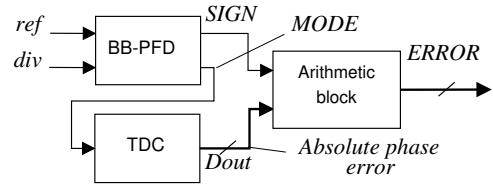

Fig. 3: Architecture of PFD

The BBPFD can be seen as a finite state automaton driven by events (rising edges) at its inputs. Inspired by [3], it is implemented similarly in ASIC and FPGA platforms.

In the ASIC-based TDC, the duration of the signal $MODE$ is quantified by a tapped delay line [4]. The sensitivity of PFD (ΔT_{TDC}^{ASIC}) is defined as one stage delay in the delay line. Since a logic gate with a wanted delay cannot be realized in FPGA, the TDC is implemented as a digital chronometer with an external clock, counting the number of the clock events included in the measured time interval. The period of the external clock corresponds to the sensitivity of PFD (ΔT_{TDC}^{FPGA}). However, the external clock is not always synchronized with the beginning of the time interval to measure. Hence, even if the time interval is below the TDC time step and if the clock event happens to be inside the interval, the FPGA-based TDC may output 1, whereas the ASIC-based TDC outputs 0. It causes a signal-correlated fluctuation with amplitude of ± 1 unit over the output code, and the sign of the fluctuation is the same as the sign of the error.
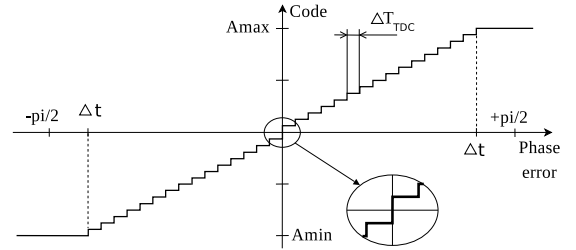

Fig. 4: Transfer function of PFD

B. Loop filter

The PI filter is implemented as the loop filter. Its transfer function is the following:

$$H(z) = \left(K_p + \frac{K_i}{1 - z^{-1}} \right) z^{-2} \tag{1}$$

where K_p and K_i are programmable gain coefficients of the proportional and integral paths respectively. The multiplier z^{-2} models the two cycles delay introduced by the two registers in the loop filter. Theoretical investigations [5] [6] provide the following specification for the coefficients: $K_p \in \{1...0.03\}, K_i \in \{1...0.00024\}$. The calculation inside the filter is achieved in fixed point arithmetic. The coefficients K_p and K_i are represented as a ratio of a programmable integer number and a power of 2 integer number:

$$K_p = \frac{K_1}{2^5}, \ K_i = \frac{K_2}{2^{12}} \tag{2}$$

where K_1 is integer in the range (0...31), and K_2 is in the range (0...$2^{12} - 1$). A constant offset 512 is added to the sum of proportional and integral values to shift the start-up value from zero to the middle of the range, which reduces the worse case frequency acquisition time by half.

The filter has up to 4 inputs (Fig. 5), which are weighted by the programmable coefficients Kw_i. The number of inputs can be regulated by programming the coefficients Kw_i to one, two, four or zero.
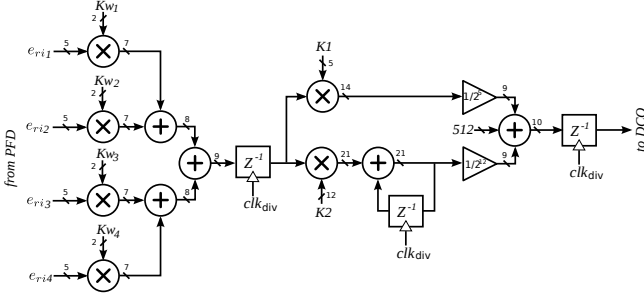

Fig. 5: Loop filter structure

The reconfiguration of the filter is implemented so as to allow all coefficients programmable. Hence, different configurations of the network can be applied and tested. The implementation of the programming is explained in Section III.

C. DCO

A DCO is a digital-to-analog converter converting a digital code into the oscillation frequency. Since analog functions cannot be implemented in FPGA (e.g., programmable delay), in this work the DCO is implemented as a pre-loaded N_c-bit counter whose overflow signal provides the output DCO clock signal. The counter uses an external clock with a high frequency ($f_{DCO\ clk}$). When the counter saturates, an output event is generated, and at the same time the counter is reloaded with the filter output code. Hence, the period of the clock generated by DCO (T_{DCO}^{FPGA}) for a certain input code C_{in} is defined as

$$T_{DCO}^{FPGA} = (2^{N_c} - C_{in}) \times T_{DCO\ clk} \quad (3)$$

where $T_{DCO\ clk}$ is the period of the external clock.

The frequency tuning step of DCO in the FPGA prototype ($\Delta f_{DCO\ FPGA}$) is defined as

$$\Delta f_{DCO}^{FPGA} = \frac{1}{T_o^{FPGA}} - \frac{1}{T_o^{FPGA} + T_{DCO\ clk}} \quad (4)$$

where T_o^{FPGA} is the nominal clock period value of FPGA-based DCO. The DCO specification (Δf_{DCO} and the central frequency f_o) defines N_c and $f_{DCO\ clk}$.

We note that in the DCO in the ASIC implementation, the code defines linearly the output frequency, instead of the output signal period in the FPGA.

D. Scaling of FPGA prototype parameters

Due to the limit of maximum frequency in FPGA device, all the frequency parameters in the FPGA network have to be scaled down. Because of the nonlinearity of the code-frequency characteristic of the DCO implemented in FPGA, a full frequency range homothety is not possible, and the downscaling is only defined for the nominal (center) DCO frequency:

$$\Delta f_{DCO}^{ASIC} : f_o^{ASIC} : \frac{1}{\Delta T_{TDC}^{ASIC}}$$
$$= \Delta f_{DCO}^{FPGA} : f_o^{FPGA} : \frac{1}{\Delta T_{TDC}^{FPGA}} \quad (5)$$

where Δf_{DCO}^{ASIC} is the frequency tuning step of the ASIC-based DCO. f_o^{ASIC} and f_o^{FPGA} are the nominal frequency of ASIC-based DCO and FPGA-based DCO, respectively. The nonlinearity of the FPGA DCO has no impact in the steady-state mode where the network is fully synchronized, since the DCO input code is virtually constant (± 1), and the DCO code-frequency characteristic can be considered as linear. However, the frequency acquisition process is not truthfully modeled by the FPGA prototype. This is not critical, since the main goal of this work is the modeling and study of the steady-state mode of the network.

For given ASIC parameters in TABLE I, if $f_{DCO\ clk}$ uses the largest frequency value of device, the other temporal parameters are calculated by using the equation (Eq.5) [7]. The result is shown in TABLE I.

TABLE I: Main characteristics of system

	ASIC	FPGA
PFD resolution (ΔT_{TDC})	30 ps	149.88 ns
DCO gain (Δf_{DCO})	200 KHz/LSB	40.03 Hz/LSB
Nominal frequency (f_o)	250 MHz	50 KHz
DCO counter frequency (f_{DCOclk}^{FPGA})		62.5 MHz

III. IMPLEMENTATION OF RECONFIGURATION FEATURES

The following features of the system are defined by programming:

- The filter coefficients. This allows a test of different loop correction strategies.
- The network topology, connectivity and weight of different links, through the programming of the filter input gains Kw_1-Kw_4 (Fig.5).

The optimal choice of the filter coefficients was addressed in theoretical work reported in [5] [6]. The possibility to test the network with different coefficients is important for the validation of these theoretical developments: this feature will be implemented on the ASIC, and the FPGA prototype allows a validation of the programming interface (cf. Section III.B).

The possibility of modifying the network topology is necessary not only for increasing the number of network topologies which can be tested, but also for starting up the network. This latter issue is discussed in the Section III.A.

A. Reconfiguration for the selection of the desired steady states

The circuit in Fig. 2 minimizes the signal "Total error", which is the sum of the phase errors of the local node with its neighbors. However, due to the cyclic (modular) nature of phase and a large number of freedom degrees in the complex system, there is a possibility to have this signal zero (actually, $n \times 2\pi$) whereas the individual errors are not zero. Moreover, such state can be a steady-state, in which the oscillators are synchronized in frequency but keeping fixed large phase errors (e.g., as shown in Fig. 6 captured by oscilloscope). Obviously, for the clock synchronization applications, such a mode is not desirable, since the phase error between the oscillators must be minimized. The problem of this architecture is its incapability of distinguishing the desired mode with all zero phase errors from undesired modes sometimes called *modelocks*. The steady state to which the system settles depends on its initial condition, which can be considered as random in a real system. Hence, the network designer should add a mechanism allowing a selection of the desired mode.

Many solutions have been proposed for this problem [1] [8]. The solution we propose is based on an artificial determination of network initial conditions from which the system settles to the desired mode. This method makes use of the network reconfigurability. The idea is to force the system to run towards the desired steady state at the start-up of system, and when the network operates near the desired steady state, the autonomy is given back to the system and the system settles down to the state at which all phase errors are zero.
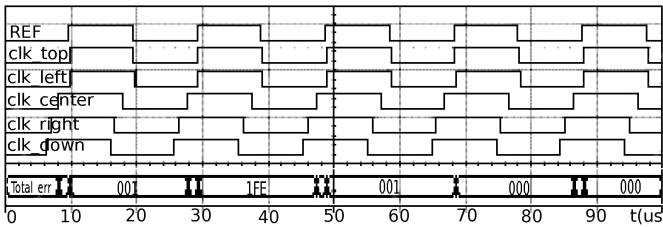

Fig. 6: One clock in network and its neighboring clocks: Undesired steady state

It has been proven that the modelocks are only possible in the PLL networks having phase error propagation loops. By consequence, unidirectional networks do not suffer from the modelock issues [1]. However, an unidirectinoal network configuration is not suitable for the oscillator synchronization, since the phase error accumulates through the network, and is amplified by the unidirecitonal PLL chains. However, the phase error provided by unidirectional configuration is small enough to determine an initial state, from which the bidirectional network can settle to the desired mode. In this way, the start-up of the network is as follows:

- The network parameters Kw_i are programmed so as to set the network into unidirectional configuration,
- After the settlement of the system, Kw_i are reprogrammed so as to set the network into the bidirectional configuration.

The switching of the connectivity is implemented by programming dynamically the coefficients Kw_i in the loop filters presented in Fig. 5.

The main challenge of this solution on large ADPLL network is the accumulation of the phase errors in the unidirectional mode: the clock generated by the nodes far from the reference (source) point has a relatively poor quality compared with clocks near the reference. By consequence, it gives a poor definition of the initial state for the bidirectional mode. For this reason, the unidirectional topology should be chosen to reduce the maximal distance of the error propagation.

Two topology of 4×4 unidirectional networks are presented in Fig. 7. To compare them, we introduce a parameter D standing for the phase error propagation distance between two clock domains, which is equal to the number of clock domain borders that the information passes through from one node to the other. Criteria of choosing the best unidirectional configuration is that the parameter D from the node at the upper-left corner (with coordinates (1, 1)) to each node in the network should be as small as possible. The topology in Fig. 7(a) uses a zigzag chain to connect all the nodes together. In this way, the value of D increases linearly as the geometry gets larger. In a 4×4 network, the distance between the last node at the end of the chain and the upper-left (which receives the reference clock) node is 15, while in a 10×10 network, the value is 99. Fig. 7(b) shows a network with a comb topology. In this case, the distance D between each node and the upper-left node is the Manhattan distance. The Manhattan distance between a node X=(X1, X2) and a node Y=(Y1, Y2) is defined as: $|x_1 - x_2| + |y_1 - y_2|$, which is the shortest distance between two intersections in a grid. In a 4×4 network, the longest distance is 6, which is between the upper-left node and the lower-right node. For a 10×10 network using this configuration, the longest distance is 18. Both configuration topologies have been implemented in FPGA and tested. The experimental results are presented in Section IV.
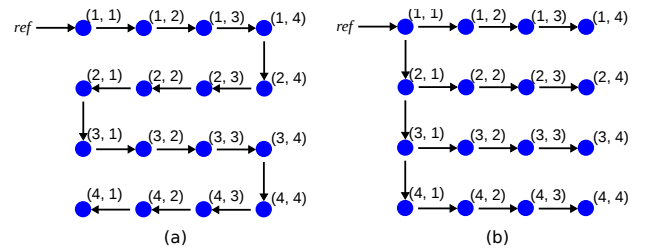

Fig. 7: Unidirectional mode topology: (a) type 1 - zigzag; (b) type 2 - comb (the arrows show the connection in the network and the error propagation direction)

B. Programming and reconfiguration of the network

In order to implement the dynamic two-step start-up technique presented above, the programming interface should allow an on-the-fly reconfiguration of the network without

suspending, which requires a simultaneous (synchronous) re-programming of all parameters in one ADPLL. Since the number of bits to be programmed is large (25 bits per node), a serial interface with two-stage parallel buffers is used for the programming sequence transmission.
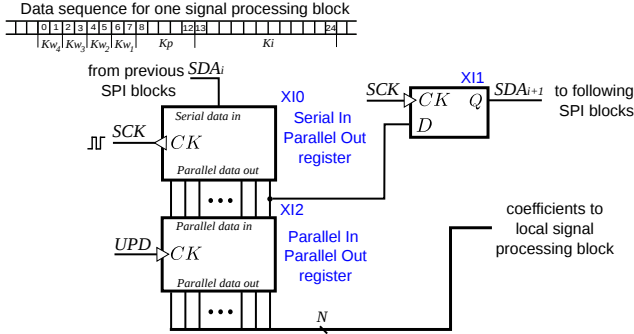

Fig. 8: Schematic of the serial programming interface (SPI)

The designed serial programming interface (SPI) in Fig. 8 is composed of a two-stage buffer (registers XI0 and XI2) and one flip-flop (XI1). The register XI0 is a serial-to-parallel converting register which shifts in the input series data on SDA_i and generates a parallel word at its outputs *parallel data out*. The actual values of the outputs of XI0 represent the bits to be programmed. However, during the read-in process, the *parallel data out* outputs have transient meaningless values. For this reason, the outputs of this register are not applied directly to the node block, but to the storage parallel register XI2. This loading is ordered by a global UPD signal. When all programming bits are sent, the UPD signal goes high. Ideally, all the network starts operating with updated parameters at the rising edge of UPD. In practice, there are skews between signal UPD arriving time at nodes in different positions of the network because of transmission delay. The maximum skew should be less than one reference clock cycle, while it is not a constraint for this low frequency FPGA prototype.

The programming interface is easily extendable by cascading (Fig. 9). The programming interfaces of two blocks are joined by connecting the last output bit of the register XI0 of one block to the SDA_i of the another block. The two blocks share the same UPD and SCK signals. In this way, for any length of the programming sequence, the interface requires only three external pads: SCK, SDA and UDP (clock, data and load signals).

IV. EXPERIMENTAL RESULTS

An FPGA emulator for a network with 100 clock generators is implemented on ALTERA CYCLONE II EP2C70 platform.

A. Effect of dynamic configuration on undesired stable state prevention

Fig. 10 demonstrates the clock signals of the main network diagonal nodes, when the network is configured in unidirectional mode. As explained in Section III.A, the phase error
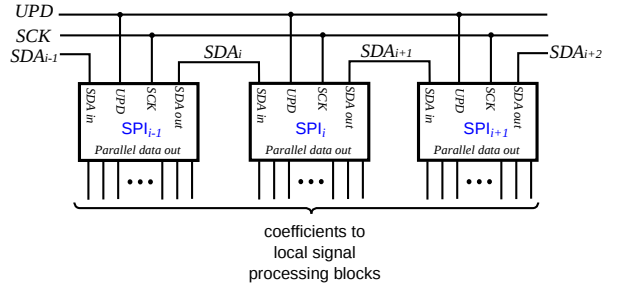

Fig. 9: Cascading the programming interfaces of several blocks

between CLK10-10 (clock generated at SCA10-10 in Fig. 1) and the reference clock REF is smaller with comb shaped configuration compared with the zigzag type.
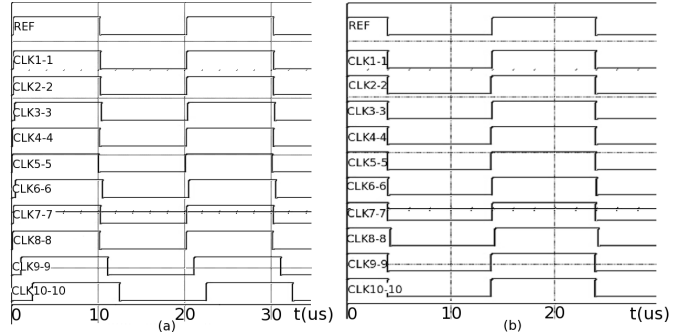

Fig. 10: Clocks in unidirectional mode at steady state: (a) type 1 - zigzag; (b) type 2 - comb

For this reason, the comb shaped topology is chosen for unidirectional mode as the first phase of the dynamic configuration process. Fig. 11 shows the four phase errors after weight coefficients (Kw_i) multiplication between node SCA7-6 and its neighboring nodes during the entire experimental process. During the period 0ms - 74ms, the network is configured as unidirectional mode, only the phase difference between the node and its left neighbor (SCA7-5) is applied to the input of the filter. Then the reprogramming is finished within 50 μs. During this time, the network still works in unidirectional mode. Since 74.05 ms, the network switches to bidirectional mode, all the four errors are taken into consideration by the filter. We can observe that during the unidirectional mode the phase error may be as large as three TDC measurement steps. However, in the bidirectional mode the maximal error is reduced to ± 2 TDC steps: that corresponds to the desired synchronized operation of the network.

B. Effect of PFD and DCO resolutions on system stability and performance

Fig. 12 demonstrates that it is not always true that a TDC with better resolution ΔT_{TDC} can improve the performance of PLL. The time step of TDC is decreased from 150 ns in Fig. 11 to 100 ns in Fig. 12(a). The PFD output is ± 4 and sometimes ± 5. If we continue to improve the resolution of
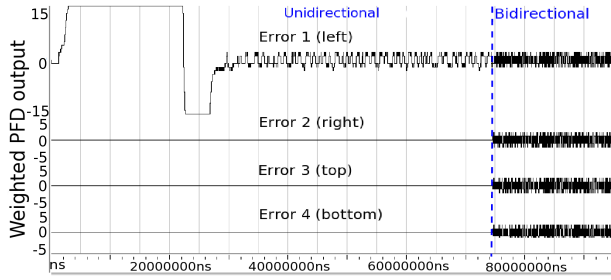
Fig. 11: Phase errors between node SCA7-5 and its neighbors (after K_{w_i} multiplication): Kp=1, ΔT_{TDC} = 150 ns, Δf = 40 Hz/LSB

TDC to 50 ns, the system is no more stable, as shown in Fig. 12(b). This corresponds to our analysis in [5] [6].
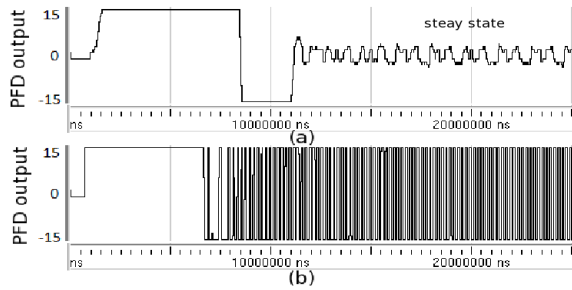

Fig. 12: Test with different ΔT_{TDC} (between node SCA7-4 and SCA7-5): (a) ΔT_{TDC} = 100 ns; (b) ΔT_{TDC} = 50 ns

Similar tests were performed for different DCO tuning step Δf. Fig. 13 demonstrates that for a certain TDC, the DCO should have a tuning step small enough to well track the reference clock.
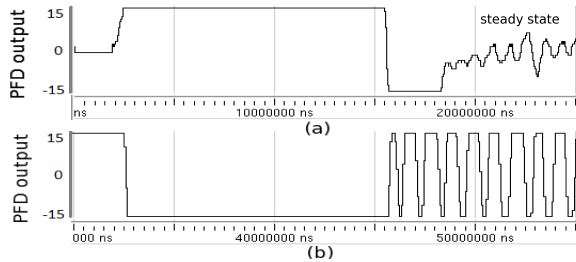

Fig. 13: Test with different Δf (between node SCA7-4 and SCA7-5): (a) Δf = 80 Hz/LSB; (b) Δf = 160 Hz/LSB

C. Effect of proportional coefficient value on system stability and performance

As analyzed in [5] [6], a larger K_p may reduce the convergence time (6.5 ms in Fig. 14(a) compared to 30 ms in Fig. 11), but might have a larger phase error in steady state (± 6 in Fig. 14(a) compared to ± 2 in Fig. 11). Moreover, a K_p too large makes the system unstable (Fig. 14(b)).
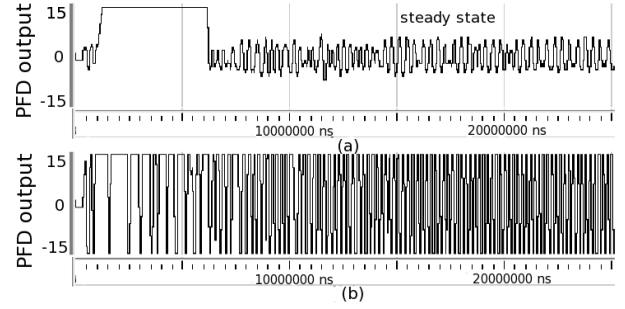

Fig. 14: Test with different K_p: (a) K_p = 2; (b) K_p = 4

V. CONCLUSION

An FPGA emulator of a reconfigurable network with coupled ADPLLs is presented in this paper. The FPGA prototyping is valuable in evaluation of proposed architecture, and in validation of reprogramming strategies which are to be implemented in the ASIC. The reconfigurability of system makes it possible to test the architecture from one single ADPLL up to a network of 10×10 ADPLLs with different topologies simply by changing the configuration without modifying the circuit. The FPGA prototype models adequately the nonlinear behavior of the ADPLL network, and allows a validation of the method aiming to select the desired synchronization mode, as well as different theoretical issues related to the studied system.

REFERENCES

[1] Pratt, Gill A and Nguyen, John, *Distributed Synchronous clocking*, IEEE transaction on parallel and distributed systems, vol. 6, n. 3, march 1995, pp. 314-328.

[2] Zianbetov, E and Javidan, M et al., *Design and VHDL modeling of all-digital PLLs*, 8th IEEE international NEWCAS conf., 2010, Montreal, QC, pp. 293-296

[3] Tierno, Jose A and Rylyakov, Alexander V and Friedman, Daniel J, *A Wide Power Supply Range, Wide Tuning Range, All Static CMOS All Digital PLL in 65 nm SOI*, IEEE Journal of Solid-State Circuits, vol. 43, no. 1, January 2008.

[4] Levine, Peter M and Roberts, Gordon W, *A high-resolution flash time-to-digital converter and calibration*, International Test Conference pp. 1148-1157, 2004.

[5] Korniienko, Anton and Scorletti, Gérard et al., *Control law synthesis for distributed multi-agent systems: Application to active clock distribution networks*, IEEE American Control Conference (ACC), pp. 4691-4696, 2011

[6] Akre, J and Juillard, Jérôme et al., *Synchronization analysis of networks of self-sampled all-digital phase-locked loops*, IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 59, no. 4, pp. 708-720, 2012.

[7] Shan, Chuan and Zianbetov, Eldar et al., *FPGA implementation of reconfigurable ADPLL network for distributed clock generation*, IEEE International Conference on Field-Programmable Technology (FPT), 2011, New Delhi, pp. 1-4

[8] Gutnik, Vadim and Chandrakasan, Anantha P, *Active GHz clock network using distributed PLLs*, IEEE Journal of Solid-State Circuits, vol. 35, no. 11, pp. 1553-1560, 2000.