Design and Characterization of a Low-Cost FPGA-Based TDC

Alessandro Tontini, Leonardo Gasparini, *Member, IEEE*, Lucio Pancheri, *Member, IEEE*, and Roberto Passerone, *Member, IEEE*

***Abstract*— We present a field-programmable gate array (FPGA) implementation of a time-to-digital converter (TDC) based on a low-cost, low-area Spartan 6 device. The converter is based on a tapped delay line model. Several implementation details are discussed with a particular focus on critical blocks such as the input stage and thermometer-to-binary decoding techniques. We implemented a tap filtering technique to improve the differential nonlinearity (DNL) of the single delay line while keeping a good LSB value of 25.57 ps with a single-shot precision (SSP) between 0.69 ÷ 1.46 LSB. Measured DNL and integral nonlinearity (INL) lie in the range between −0.90 ÷ 1.23 and −0.43 ÷ 2.96 LSB, respectively. Measured DNL and INL lie in the range between −0.90 ÷ 1.23 and −0.43 ÷ 2.96 LSB, respectively. We then implemented an interpolating TDC to overcome the limitations of a single delay line in terms of linearity and measurement range. The interpolating TDC uses the sliding scale technique, where the time interval to be measured is asynchronous with respect to the FPGA clock, achieving DNL and INL in the range −0.072 ÷ 0.070 and −0.755 ÷ 0.872 LSB. SSP is in the 1.096 ÷ 2.815 range. Moreover, we present a novel comparison between the DNLs obtained with two different methods: statistical code density test and using a finely controlled delay source. Finally, we present the results of a Monte Carlo simulation used to investigate the effects of nonlinear propagation of the signal through the delay line.**

***Index Terms*— Differential nonlinearity (DNL), field-programmable gate array (FPGA), jitter, Monte Carlo simulation, time-to-digital converter (TDC), thermometer-to-binary decoder.**

I. Introduction and Related Work

THE high time resolution a time-to-digital converter (TDC) can provide is needed in many scientific areas, from high-energy and medical physics [1]–[3] to time-of-flight ranging [4] and fluorescence lifetime spectroscopy in chemistry and biology [5]. The required range and resolution, and thus the specifications of the TDC, are very strongly application dependent. CMOS application-specified integrated circuits (ASICs) offer the flexibility to customize TDC-based systems according to the specific system requirements, providing an ideal platform to implement TDCs with single-shot precision (SSP) down to 1 ps [6] as well as very large TDC arrays [7], [8]. The implementation of a dedicated ASIC, however, requires a relatively long time that is not suitable in all the projects. For many systems requiring TDC, the fast deployment and reasonable costs offered by field-programmable gate array (FPGA)-based solutions are very often a valid alternative.

FPGAs have their strengths in reconfiguration, design portability, fast prototyping phase, and relatively low budget depending on the board model. These features pushed the research interest toward TDCs based on FPGA devices, even if there are obvious obstacles when compared to an ASIC design in which the TDC is specifically designed around a target technology.

There are several FPGA-based TDC architectures reported in the literature. One of the very first implementation is described by Kalisz *et al.* [9] using a Vernier delay line in a 0.65-μm QuickLogic FPGA, reaching a resolution of 200 ps. A different approach, proposed by Balla *et al.* [10], makes use of multiphase clocks in order to reach a precision down to one fourth of the system clock period. However, the most common technique employs a tapped delay line model using dedicated carry lines within the FPGA fabric. Dedicated carry lines provide a built-in tapped delay line with CMOS-gate propagation delays down to some tens of picoseconds. One of the very first work based on this principle is presented by Song *et al.* [11], using both Altera and Xilinx devices and reaching precisions of 65 and 46.2 ps, respectively. In 2009, Favi and Charbon [12] developed a 17-ps resolution TDC based on a TDL model in a 65-nm Xilinx FPGA. Similarly, the work of Fishburn *et al.* [13] presents a precision of 19.6 ps. More sophisticated approaches have been developed in order to reduce the nonlinearity of the tapped delay line. Shen *et al.* [14] implemented multiple delay lines in order to average the final measure to improve the time resolution. Another technique reduces the differential nonlinearity (DNL) directly on chip using histogram-based compensation techniques (as in [15]) and with the addition of multiple-phase clocks as in [16]. The aforesaid technique, however, requires a large amount of FPGA resources, thus limiting their implementation to high-density devices. Won and Lee [17] present a tuned delay line approach that analyzes the behavior of the delay line and selects a preferential sampling pattern of the carry line that is unique for each FPGA class or model. A similar approach is used in [18], where the delay line behavior is studied through the post place-and-route timing report, and taps are sorted by increasing delays accordingly.

Manuscript received November 3, 2017; revised December 23, 2017; accepted January 3, 2018. Date of publication January 8, 2018; date of current version February 13, 2018.

A. Tontini and L. Gasparini are with Fondazione Bruno Kessler, 38123 Trento, Italy (e-mail: tontini@fbk.eu; gasparini@fbk.eu).

L. Pancheri is with the Department of Industrial Engineering, University of Trento, 38122 Trento, Italy (e-mail: lucio.pancheri@unitn.it).

R. Passerone is with the Department of Information Engineering and Computer Science, University of Trento, 38122 Trento, Italy (e-mail: roberto.passerone@unitn.it).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNS.2018.2790703

In this paper, we contribute to the field of TDCs in several ways. First, most of the FPGA-based TDCs reported in the literature are based on high-level, high-gate count FPGAs. In this paper, we show that a low-cost, low-gate count Xilinx Spartan 6 device can be used to achieve high resolution and high dynamic range with nonlinearity metrics that are in line or (significantly) better than those reported in the literature using more expensive hardware. We study and characterize all stages of an interpolating TDC. In particular, we propose a new input stage that guarantees synchronization, and evaluate the impact on the performance of two types of thermometer-to-binary decoders, showing their effect on nonlinearity and monotonicity. We then validate the traditional code density test method used to measure nonlinearities (DNL, integral nonlinearity (INL)) by comparing the results with the same measure obtained from the conversion curve through the injection of finely controlled delays. A Monte Carlo simulation is used to delve into the relationship between the intrinsic nonmonotonicity of the delay line and different conversion techniques. Moreover, we analyze the effect of system jitter on the linearization of the conversion curve. Eventually, we evaluate the TDC using a time-correlated single-photon counting (TCSPC) application, to measure the fluorescence lifetime decay and compare it with a professional commercial instrument.

This paper is organized as follows. Section II gives a functional overview of the system, discussing the role of each component. Then, Section III explores alternative implementations and provides a characterization of the performance of the delay line and of the measurement method. Finally, Section IV discusses the performance of our TCSPC setup. Conclusive considerations are provided in Section V.

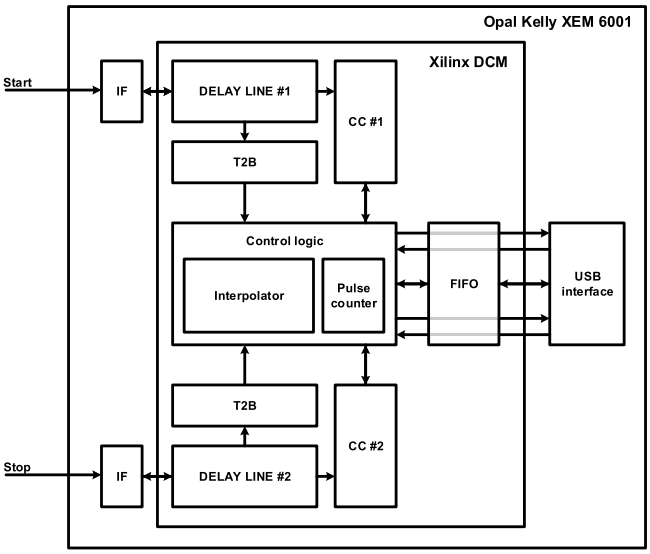

Fig. 1. System block diagram.

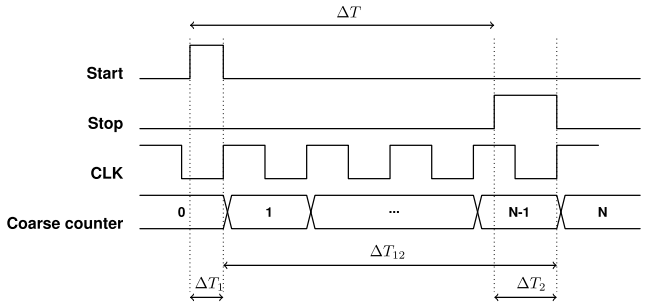

Fig. 2. Principle of operation. The time measure is composed of three elements: two fine measures, obtained with the delay line, plus a coarse grained measure to arbitrarily extend the measurement range.

II. Functional Overview

The purpose of a TDC is to measure the time interval between two events, identified by the activation of a start and a stop signals, respectively, and to provide the result in digital format. This can be easily accomplished by starting and stopping a counter, which counts the number of clock periods that elapse between the two events. Since the events are asynchronous, this technique yields a resolution that is no better than the clock period, and therefore, fairly limited (few nanoseconds), especially using low-cost devices. A much higher resolution can be achieved by running the input signal, as soon as it is received, through a fast tapped delay line, and then sampling the position that is reached when the stop signal is asserted. Delay lines can be constructed with simple gates with small delay, giving a much more fine grained measure.

The maximum delay line length, which can be reliably constructed with the FPGA used in this paper, puts a serious limitation on the available measurement range. This is incompatible with many applications, such as TCSPC, where ranges of some tens of nanoseconds are needed in order to accurately estimate the exponential decay. To overcome this limitation, we implemented an interpolating TDC, according to the Nutt interpolation method [19], which combines the use of delay lines for resolution and of counters to extend the range. The interpolating TDC has been implemented following an asynchronous model using two different delay lines for the start and stop signals, respectively. Our implementation is shown in Fig. 1. The system is composed of two input sections, which separately handle the two event signals, and a controller, which acquires and processes the data and communicates with a host PC. The principle of operation is depicted in Fig. 2. Delay line #1 measures the time difference ΔT_1 between the start signal and the next active edge of the clock. Clock counter #1 (CC1) is then activated to count the remaining n clock cycles, until the stop signal is received. Because the stop signal may arrive in the middle of the clock period, delay line #2 measures the time ΔT_2 between the stop signal activation and the next active clock edge, which also disables CC1. Thus, the overall time interval ΔT is given by $\Delta T = \Delta T_1 + n \cdot T_{\text{clk}} - \Delta T_2$, where T_{clk} is the clock period.

The advantage of an asynchronous design relies on the so-called *sliding scale technique*, introduced by Cottini *et al.* [20] and used in a Vernier interpolating loop with a 17-ps resolution [21]. The sliding scale technique principle of operation is depicted in Fig. 3, where the same time interval, ΔT, is measured in different regions of the start and stop delay lines. By doing so, nonlinearities are averaged across multiple observations, thus overcoming the limitations of a single delay line design.

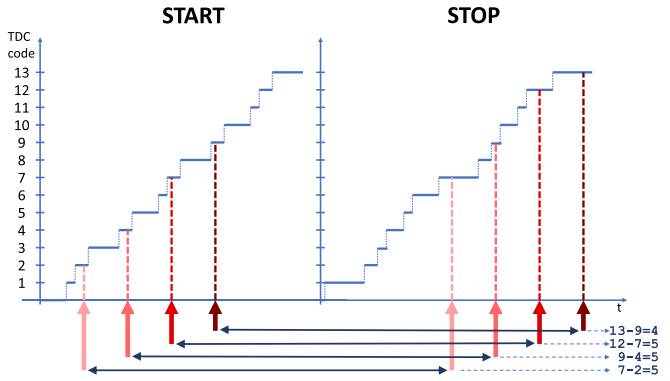

Fig. 3. Asynchronous time measure using the sliding scale technique. The same time interval can be measured over different regions of the start/stop channel delay lines, improving linearity.

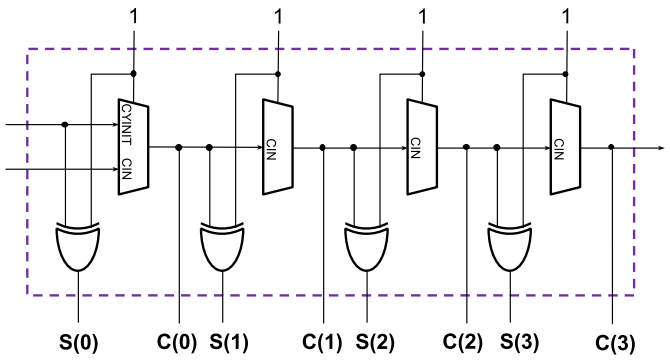

Fig. 4. CARRY4 primitive simplified reference schematic. The signal propagates through the multiplexer gates. The output can be taken from both the multiplexer output or the XOR gate output.

In addition to the delay line and the counter, the two acquisition sections include an input stage (IF) that shapes the input according to our needs, and an encoder (T2B) that translates from the supposedly thermometer representation of the delay line to a more convenient binary format. Clock counter #2 (CC2) is needed to handle the cases where two subsequent start signals occur, for example, when an event detection is missed in the stop channel. In such cases, it is useful to have a backup counter ready to start, rather than waiting for CC1 to be stopped, reset, and started again. The control logic performs the additional processing required to implement interpolation, and stores the measurement data in an FIFO queue, which connects to the host PC through a USB 2.0 interface. In addition to the time intervals, the system uses a pulse counter to collect statistics about the number of start and stop events per second, which are periodically sent to the host PC and are useful in certain applications, such as in TCSPC experiments in order to monitor the signal intensity. The target device is a Xilinx Spartan 6 XC6SLX16-2 FPGA, embedded in an Opal Kelly XEM6001 module. FPGAs are built according to a repetitive, cellular pattern whose fundamental block is the *slice* (according to Xilinx terminology). Among other things, slices contain the *fast carry logic*, named CARRY4, which we employ to build the tapped delay line. A simplified reference schematic is shown in Fig. 4, where only the signals of interest for the purpose of the delay propagation are depicted. Each CARRY4 primitive has four delay elements, each providing two distinct outputs: the carry output, from the multiplexer gate (**C**), and the sum output, from the XOR gate (**S**). The time resolution of the FPGA-based TDC is given by the time delay introduced by a single element, usually in the order of tens of picoseconds. The designer is free to select which output to consider. In the rest of this paper, we will denote the choices using a pattern of letters **C** or **S**, while we will denote the case in which a particular output is ignored using the letter **N**. For example, the pattern **NCCC** means that the first output is ignored, while the remaining outputs are taken from the carry output.

Because in the worst case a full clock period may elapse between an input event and the next active edge of the clock, the tapped delay line must introduce a delay that is greater than or equal to the FPGA clock period. This can be accomplished by chaining several CARRY4 primitives to construct a sufficiently long line. Due to routing restrictions, we can only connect CARRY4 blocks vertically within the FPGA fabric to make a chain, thus limiting the maximum allowable delay line length. The total delay for a single chain in this case is approximately 4.5 ns, which bounds the minimum system clock frequency to be at least $\simeq$220 MHz. We have therefore developed a relatively simple design, making use of pipelining where needed, to satisfy these timing constraints. In the end, the system runs at 230 MHz in order to maintain some margin on the clock period and achieves a sampling rate of 115 Msample/s. The delay line is made of 240 elements, the maximum allowable by the FPGA model we used in our work.

Regarding the thermometer-to-binary decoder, we have developed and tested two approaches. The first is the classic *leading one detector*, which tracks the position of the last propagated one in the delay line and generates the corresponding 8-bit value [18]. The second solution, which we refer to as the *ones-counter*, counts instead the overall number of ones in the delay line. We compare the performance of these decoding techniques in Section III-C. Both decoders are implemented in a 32-stage pipeline, in order to deal with the high system clock frequency of 230 MHz. Other approaches, such as a priority encoder or a mux-based decoder, were not suitable for this frequency due to the large usage of combinatorial logic.

III. Implementation and Characterization

In this section, we analyze the design choices and characterize the performance of the different parts of our system. In particular, we first characterize a single delay line, looking at the input stage behavior, the DNL and how to measure it, then we discuss some common sources of nonlinearity and the relation between conversion mechanism and intrinsic nonmonotonicity. Section IV will instead look at the interpolation method.

A. Input Stage and Delay Line Placement

Before being injected into the delay line, the asynchronous input signal is processed by the input stage, which plays two

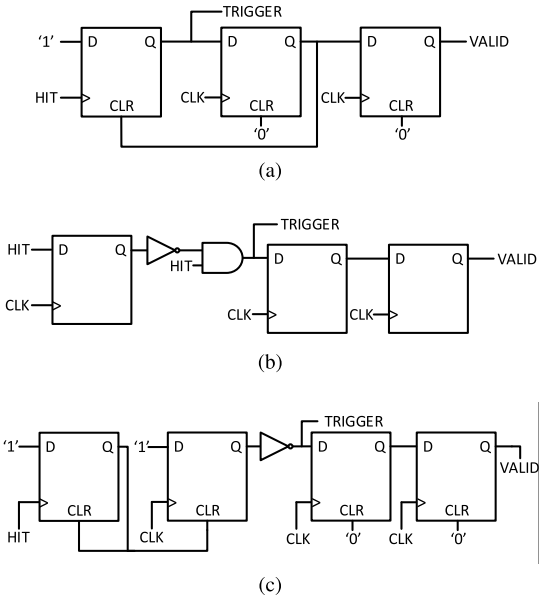

Fig. 5. Different topologies for the input stage based on both combinatorial and sequential logic that shape the asynchronous input signal to be active for one clock period and generate a valid flag for the thermometer converter.

fundamental roles. First, it shortens the input pulse to be active only from the time it is received up to the next useful clock edge. This way, the delay line performs a single measure even if the input signal remains asserted for a long time. In addition, possible noise (intended as multiple transitions per pulse) in the input signal is filtered out. Second, the input stage generates a synchronous trigger, called `VALID`, which denotes that a measure is available and starts the thermometer-to-binary conversion.

Several circuit topologies can be employed for the input stage. Sample implementations make use of multiple discrete flip-flops and logic gates that were manually placed close to the first CARRY4 primitive. Their schematic is shown in Fig. 5(a)–(c). Here, the `HIT` signal corresponds to the input event, `TRIGGER` is what is fed to the delay line, while the `VALID` signal marks the beginning of the conversion, which in our case takes place one clock cycle after the acquisition. In particular, topology Fig. 5(c) is the one proposed by the Delft Basic FPGA TDC design.[1] In most cases, the input event acts as a clock signal for a flip-flop, which transitions to the active state (the input is the constant 1), until it is cleared through an asynchronous command. The other flip-flops in the design are instead controlled by the system clock. We have tested the input filters by injecting trains of precisely delayed pulses, observing their behavior over the full range of the delay line values. Pulses are generated by a Keysight 81150A pulse signal generator triggered by the FPGA with a clock-synchronous periodic signal and fed to the FPGA through a fast global clock input pin. The results for these topologies are not satisfactory. In particular, we have observed a worsening of the SSP and a reduction of the total dynamic range of the delay line with respect to our new topology. These issues are due to routing delays between the connections of the input

[1] http://cas.tudelft.nl/fpga_tdc/TDC_basic.html

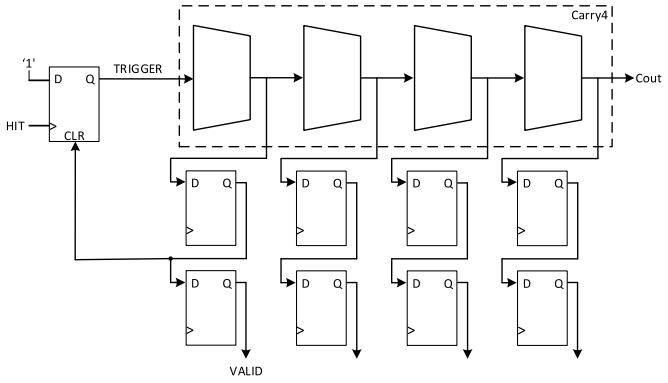

Fig. 6. Input stage currently implemented.

stage flip-flops and logic gates, resulting in a real behavior substantially different from the ideal one. The latter issue in particular is caused by race conditions that occur between the trigger signal sent to the delay line and the clock signal (these two are obviously not synchronous), causing the valid signal to be asserted during the wrong clock period, when the delay line has been reset.

To overcome these problems, we have therefore designed the new topology as shown in Fig. 6, which behaves close to the ideal case. The main improvement is that the signals needed to reset the delay line and to assert the valid signal are extracted from the delay line itself, thus removing most sources of poor synchronization. In particular, the first bit of the thermometer code is used to generate the reset (`CLR`) signal that clears the flip-flop (from the first sampling stage) and the `VALID` signal for the conversion (from the second sampling stage). This method is resilient against any desynchronization problem, since the `VALID` signal that enables the thermometer-to-binary conversion is raised if and only if a signal propagates into the delay line, and is delayed through the same pipeline stages as the data. Moreover, the resource usage is highly optimized, since only one extra flip-flop is employed besides those used by the sampling stage.

As for the input stage, also the delay line was manually placed starting from a given FPGA position. As mentioned in Section II, due to our restrictions in terms of the maximum delay line length, we placed the first CARRY4 element starting from the bottom part of the FPGA. The delay line is then developed over a vertical line, using the dedicated fast routing between each CARRY4 element. We therefore set compiler constraints in terms of both speed, to achieve the required 230-MHz clock frequency, and of placement to lock the positions of the critical elements. As a result, performance across each compile run was stable, since the most critical blocks are always constrained and routed in the same position.

B. DNL Characterization

The DNL compares the size of each step in the digital output with the nominal resolution, i.e., 1 LSB. One way to tune the linearity is to analyze the Xilinx post place-and-route static timing report, where tap-to-tap delays are reported according to the selected FPGA model and speed grade [18]. In addition,

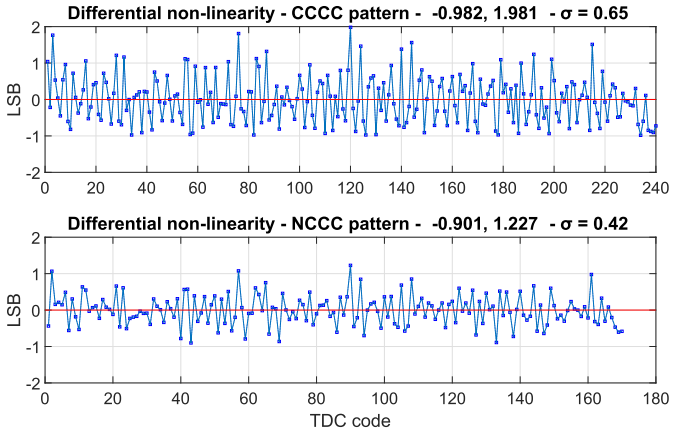

Fig. 7. DNLs improvement with filtering method. With the **CCCC** pattern, extreme values are in the $-0.98 \div 1.98$ range, with the **NCCC** pattern, extreme values are reduced in the $-0.90 \div 1.23$ range. The standard deviation, σ, improved from 0.65 to 0.42 (35.4%).

it is possible to tune the output of each CARRY4 primitive and find a preferential pattern (in terms of **C–S**) that improves the linearity of the delay line [17]. For example, the best output pattern for a Spartan-6 XC6SLX45T-FGG484-3C was found to be **SCSS** [17]. We tested all 16 combinations of **C** and **S** patterns for our FPGA, and discovered that the best output pattern is given by the classical **CCCC**. At the same time, we noticed that once taps are sorted according to the Xilinx delay model [18], the total delay of certain neighboring taps differs only by very small amounts (1 ps). This means that not all taps provide useful data. We therefore implemented a downsampling approach, in which one out of four taps per CARRY4 primitive is ignored. Downsampling improves linearity, but obviously reduces the nominal time resolution, since fewer taps are used for the same total delay. From this perspective, the term "resolution" defined in the literature as time range/full scale is somehow deceptive, as it does not take into account missing codes and other strong nonlinear behaviors. Nonetheless, our method is less aggressive than a $2\times$ or $4\times$ downsampling, where the delay line resolution is degraded by the corresponding factor. Again, we measured all possible downsampling patterns, and found the best to be **NCCC**. In this case, the LSB increases from 19.26 ps for the full case to 25.57 ps for the downsampled case. However, a better DNL is much more desirable than a high *raw* resolution. Fig. 7 shows the measured DNL for both the **CCCC** and the **NCCC** cases. The results for the **CCCC** case are in line with those reported in the literature, whereas the **NCCC** case improves the behavior of the single delay line linearity even if compared to higher level FPGA TDCs [12], [13]. The extreme values of the DNL are reduced from 1.93 to 1.23 for the positive values, and from -0.98 to -0.90 for the negative values. For the **CCCC** pattern DNL, 19 bins are greater than 1, while 12 are lower than -0.9. Using the **NCCC** pattern, only three bins are greater than 1 and just one bin is slightly lower than -0.9. Thus, at the expense of a slight decrease in time resolution, nonlinearity figures are improved with respect to the standard case, in which all taps of the delay line are considered.

TABLE I
TDC PERFORMANCE TABLE

Metric	Value	Units
Clock frequency	230	[MHz]
LSB	25.57	[ps]
Dead time	8.69	[ns]
Measurement rate[1]	16	[MSa/s]
Size[2]	415/2278	[Slices]
Power[3]	131	[mW]
Single delay line		
SSP	0.69 ÷ 1.46	[LSB]
DNL	-0.90 ÷ 1.23	[LSB]
INL	-0.44 ÷ 2.96	[LSB]
Interpolating TDC		
SSP	1.096 ÷ 2.815	[LSB]
DNL	-0.072 ÷ 0.070	[LSB]
INL	-0.755 ÷ 0.872	[LSB]

[1] According to the PipeOut measured performances in [24].
[2] Number of occupied slices over the total available amount in the XC6SLX16.
[3] Estimated through the Xilinx XPower analyzer, result of the combination of 55 mW of quiescent power and 76 mW of dynamic power.

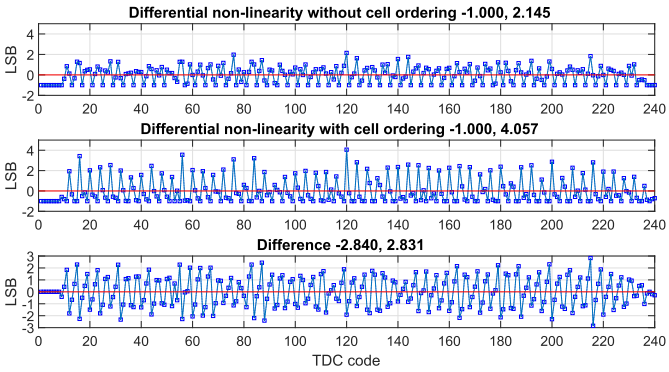

Fig. 8. DNLs comparison plot: a leading one detector is not resistant against taps ordering. The maximum and minimum values for each plot are reported in each title.

The INL is computed as the cumulative sum of the DNL and it has not been graphically reported for the sake of compactness. INL values for both start and stop delay lines are reported in Table I.

C. Decoder Performance

The way the thermometer data are encoded into a binary value has an impact on the performance of the TDC system. In particular, we have assessed the performance of both the *leading one detector* and the *ones-counter*. As suggested in the literature, we have combined the decoding technique with the process of *tap sorting*, in which taps are ordered according to the Xilinx post place-and-route timing report [18]. In all cases, we have used the **CCCC** pattern without downsampling to have the largest dynamic range and time resolution. Fig. 8 shows the measured DNL for the leading one detector, *without* (top) and *with* (middle) tap sorting, and their difference (bottom). Clearly, because this decoder selects the highest active tap, a different order results in a different converter

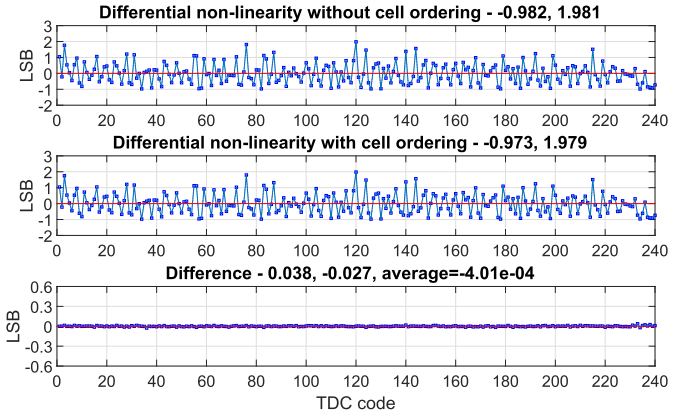

Fig. 9. DNLs comparison plot: using a ones-counter approach automatically sorts taps. The maximum and minimum values for each plot are reported in each title.

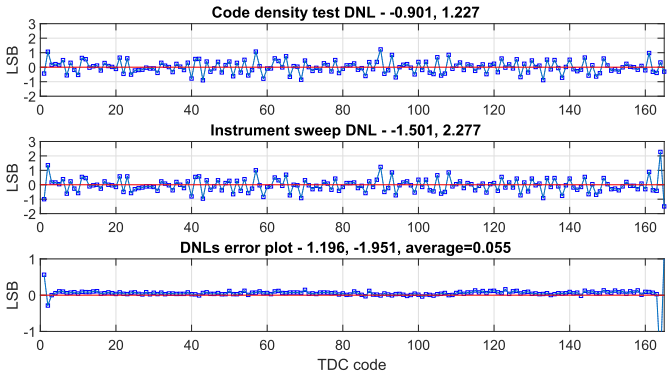

Fig. 10. DNLs comparison plot. The almost zero difference between the two DNLs is the proof of the reliability of the statistical code density test against a classical and more complicated approach.

behavior. On the other hand, Fig. 9 shows the same data for the ones-counter. Results are perfectly coincident in this case, as if they were different realization of the same process. In other words, because it simply counts the number of active taps, the ones-counter automatically sorts the delay line output. It is clear, then, that any operation of tap sorting makes sense only if a leading one detector-style decoder is implemented [18]. Effectively, the ones-counter makes this step unnecessary.

D. DNL Measurement Method

It is a common practice to compute the DNL of a TDC from a statistical code density test. As the name suggests, a train of randomly generated pulses are fed into the delay line, thus producing values across all the available dynamic range. The histogram of the distribution of the measured values is then used to estimate nonlinearity metrics. The DNL, in particular, is computed as the relative "size" of an output code, intended as the number of times the code is generated compared to the average. A more direct way to compute the "size" of an output code is to start from the *conversion curve*, which gives for each input time interval the corresponding output code. The difference between the highest and lowest time interval of each code, relative to the ideal size, corresponds to the nonlinearity. This approach is typically used in the field of ADCs, since it is quite easy to build a fine-controlled voltage source to characterize the converter. On the other hand, fine-controlled time sources are less common. Having one such controlled time source available, we decided to compare the DNLs obtained with the two methods in order to assess the reliability of the statistical code density test.

Fig. 10 shows the result of our comparison. Both measures were obtained using an **NCCC** pattern with a ones-counter thermometer-to-binary decoder, in order to have an automatic tap sorting, as in Section III-C. The first DNL plot was obtained through the classical statistical code density test. To have a truly random source, pulses come from a single-photon avalanche diode illuminated by ambient light. The DNL is computed as follows:

$$\mathrm{DNL}_i = \frac{n_i - \overline{n}}{\overline{n}} \tag{1}$$

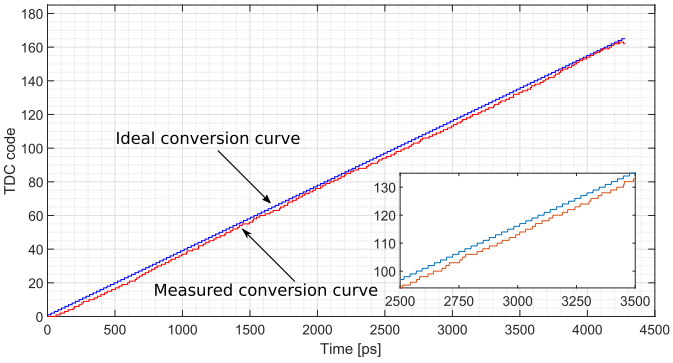

Fig. 11. Conversion curve comparison for the **NCCC** pattern. The measured curve has been shifted down by 2 LSB for better visualization.

where n_i is the number of counts for the ith histogram bin and $\overline{n}$ is the histogram average height, i.e., the total number of counts divided by the number of bins.

To extract the DNL with the second methodology, we first measured the conversion curve using a Keysight 81150A signal generator. An FPGA clock-synchronous signal is provided to the external trigger input of the Keysight 81150A generator, which was remotely controlled through a MATLAB program to sweep the entire range of delay values in steps of 1 ps. For each 1-ps step, 8192 measurements are taken and the median is considered as representative of that delay value. The *measured* conversion curve is shown in Fig. 11 superimposed with the *ideal* conversion curve for visual comparison. The second DNL plot of Fig. 10 was then obtained from the conversion curve with the classical size computation, according to

$$\mathrm{DNL}_i = \frac{(T_{i+1} - T_i) - LSB}{LSB} \tag{2}$$

where

T_{i+1} transition point between code i and code $i+1$;
T_i transition point between code $i-1$ and code i;
LSB least significant bit value, which is equal to 25.57 ps for the **NCCC** pattern.

Fig. 10 clearly shows that the DNL obtained through the code density test closely matches the DNL obtained through

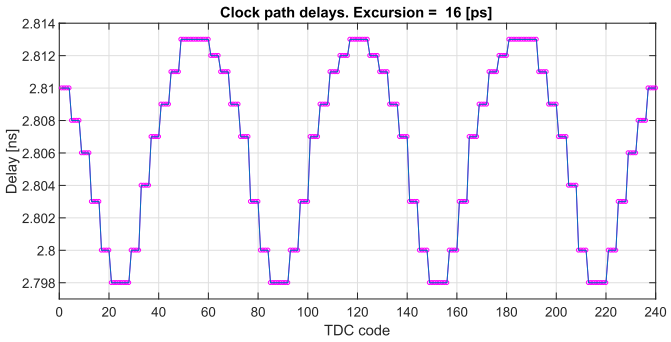

Fig. 12. Clock path delay for the FPGA used in this paper. Each cluster of dots represents four flip-flops within a SLICE, served by a clock terminal branch.

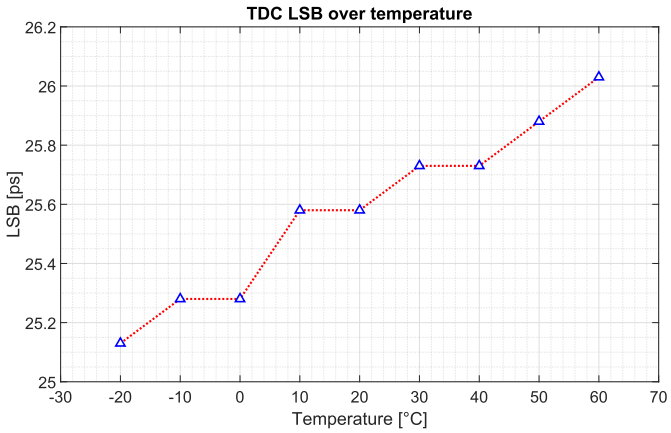

Fig. 13. LSB value as a function of environment temperature.

the instrument delay sweep method. This is a proof of the reliability of the statistical code density test. Outliers (smallest and highest values) are not considered, due to the delay wrapping around the clock edge that produces both the highest and lowest values, thus distorting the median statistic.

E. Sources of Nonlinearity

One source of distortion in FPGA-based TDCs is represented by the clock distribution skew across the flip-flop chain that samples the state of the delay line. This could lead to "bubbles" in the thermometric code, that are to be solved either by software or hardware processing. This issue affects in particular large FPGAs, where the clock signal has to travel long distances to reach all elements of the delay line. In our work, given the small size of the FPGA, this effect can be substantially neglected. To confirm this, we analyzed the Xilinx post place-and-route report, where the clock distribution delay is reported for each SLICE, and is shown in Fig. 12. The data are consistent with the actual clock routing scheme obtainable from the Xilinx FPGA Editor tool, which has not been graphically reported for the sake of compactness. The maximum excursion we have in our FPGA is only 16 ps, which is lower than our LSB ($\simeq$25.5 ps). The main technique to reduce this issue is to decrease the size of the delay line, at the expense of increasing the system clock frequency. This can be easily accomplished in high-level/high-speed FPGAs, but it is more difficult (even impossible) to do with small entry-level FPGAs. For comparison, we targeted a high-level Kintex-7 XC7K160T FPGA and compiled a design with the same 240-element delay line as we implemented in our work. The Xilinx post place-and-route static timing report shows a maximum excursion in the clock path delay of 55 ps, that is approximately five times larger than the LSB value obtainable with the Kintex-7 device. This larger value is consistent with the different routing scheme employed by this specific FPGA model, since many more slices are served by the same clock routing branch. In fact, in our FPGA (Spartan-6 XC6SLX16), the clock is distributed across four clock regions using four branches. Each branch delivers the clock signal to 16 slices, starting from the center to eight slices on the left and eight slices on the right. Conversely, in the Kintex-7 XC7K160T, the clock is distributed across five clock regions using five branches, with each branch serving 50 slices, 25 on the left and 25 on the right. Effectively, the clock routing scheme of a smaller FPGAs is more efficient than on bigger FPGAs. For reference, a 960-element delay line in the large FPGA results in a skew of 357 ps.

Another source of distortion is represented by the drift of the LSB value due to temperature variations. In order to estimate this variation, we measured the full-scale value of the delay line in a temperature-controlled environment by injecting FPGA-synchronous delays to reach the full-scale range. This analysis is reliable only if the FPGA system clock is stable against the temperature variation. To verify this, we measured the period of a clock-synchronous signal at $1/16$ of the original clock frequency under extreme temperature conditions of -20 °C and $+60$ °C through a LeCroy WavePro 7200 oscilloscope. At -20 °C, the measured mean period is 69.559 ps, while at $+60$ °C, we obtained a mean period of 69.561 ps. The FPGA clock can therefore be considered sufficiently stable for a temperature-dependent analysis. Fig. 13 shows the LSB drift for a wide range of temperatures. The LSB peak-to-peak variation is approximately 1 ps. This result is in line with the literature [13], [22], where the same phenomena have been analyzed.

F. Effects of Nonlinear Propagation

According to the result of the Xilinx post place-and-route report shown in Fig. 14, a delay line implemented using a **CCCC** pattern shows a nonmonotonic behavior, since delay differences of subsequent taps are negative. However, the intrinsic nonmonotonicity of the delay line could be hidden by certain conversion techniques. Both the leading one detector and ones-counter approaches are able to automatically solve the monotonicity issue. In the leading one detector case, the output value is that of a classic thermometer-to-binary conversion, i.e., the position index of the last "1." Therefore, even if the underlying characteristic is nonmonotonic, the last "1" in the thermometer code hides previous bits still at "0." The end result is that of a monotonic behavior with missing codes, that are hidden by this particular conversion strategy. In the ones-counter approach, the output value is

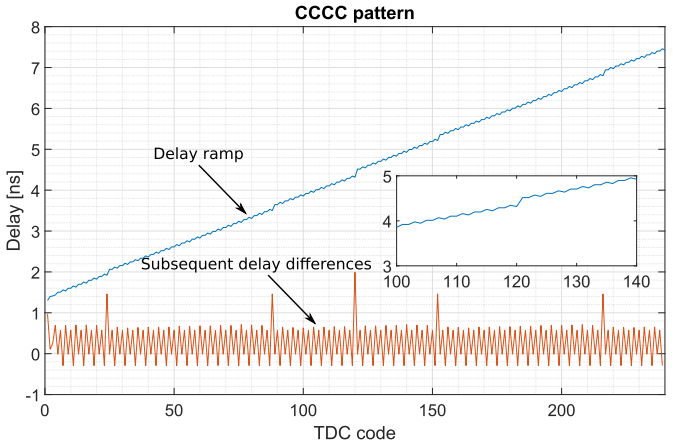

Fig. 14. Xilinx model delay behavior for CCCC pattern. Some delay differences are negative, symptom of a nonmonotonic curve.

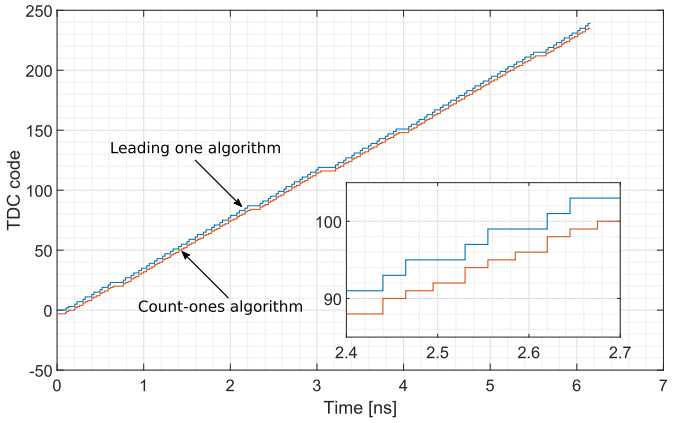

Fig. 15. Simulation results of different conversion strategies. The intrinsic nonmonotonicity leads to missing codes using a leading one detector, which are avoided using a ones-counter approach.

simply the overall number of switched bits. Also in this case, the nonmonotonicity is automatically solved. However, due to this particular conversion strategy, all taps are considered, so no missing codes are produced, as opposed to the leading one detector.

To support these considerations, we run a Monte Carlo simulation using the Xilinx models of the nonmonotonic **CCCC** delay line of Fig. 14. We simulated increasing delays of 5 ps, in order to have a consistent number of measures for each output code. The result of our simulations are reported in Fig. 15.

In particular, the inset clearly shows that the conversion curve obtained using a leading one detector has fewer steps (denoting the presence of missing codes) than the conversion curve from the ones-counter approach. This is in line with the experimental results from the code density tests of Figs. 7 and 8 (top), where the ones-counter approach (that has been used in the final implementation) results in a better DNL than the leading one detector.

We then run additional simulations to investigate the interaction between nonmonotonicity, the coding scheme, and jitter. More specifically, we simulated a coding scheme that returns

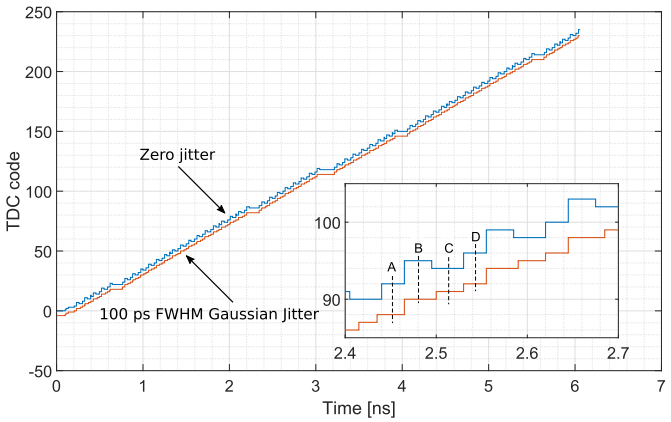

Fig. 16. Simulation results of the popping one conversion technique. Tracking the last switched bit in the delay line is not resistant against nonmonotonicity. However, monotonicity can be hidden by system jitter.

the index of the *latest* tap, time-wise, to switch to value "1" in the delay line. In other words, we track the last "1" in the time dimension, rather than in the spatial dimension as for the leading one detector, so that conversion is affected by the nonmonotonic behavior of the delay line. We will denote this encoding as *popping one*. In particular, we wish to assess the impact of jitter on the conversion curve. Similar to the previous case, we simulated increasing delays of 5 ps, and considered the median of a set of 8192 measures for each step, with and without jitter. For the purpose of simulation, there is no difference whether the jitter is added to the FPGA clock rather than to the input signal. However, in a real implementation, it is desirable to have a low jitter FPGA clock. The results are reported in Fig. 16.

In the ideal case where the system jitter is zero, the simulated conversion curve is nonmonotonic. Interestingly, the inclusion of a Gaussian jitter with a full width at half maximum, $\text{FWHM}_{\text{jit}} = 100$ ps, makes the conversion curve monotonic. Fig. 17 shows the distribution of the outcomes of the 8192 experiments for the taps from 88 to 100 (including points A, B, C, and D of Fig. 16), as the input increases its delay from 2.45 to 2.54 ns. The effect of jitter, considering the median of a number of measures per TDC bin, turns the pattern monotonic. In particular, by looking at the patterned bins of Fig. 17, the output pattern is now 92-94-95-96, that is monotonic.

As the popping one scheme is not practically feasible, the result lead us to investigate the effect of jitter on the other conversion schemes. In particular, referring to the leading one approach, we noted that jitter could have a beneficial effect on the linearization of the conversion curve, at the expense of the SSP. We simulated a Gaussian jitter with FWHM_{jit} from 4 up to 512 ps. Fig. 18 shows the conversion curve obtained by simulating increasing delays of 5 ps, considering 8192 measures per step. For each point, we consider the average of the output codes of the delay line. Because the jitter makes the output toggle between neighboring codes, by collecting several measures and taking the average, we are able to recover the skipped codes. Naturally, for the jitter to have a

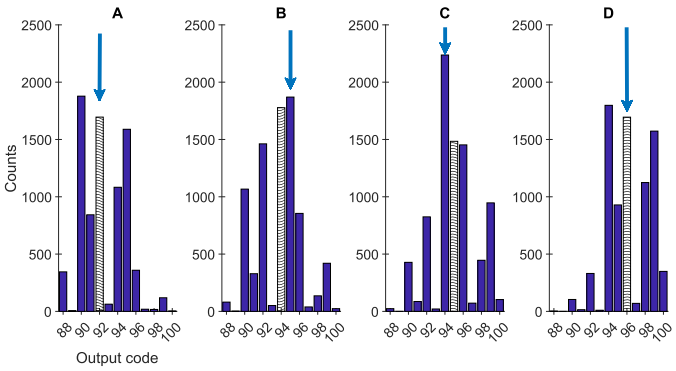

Fig. 17. Detailed analysis of the effect of jitter to the nonmonotonicity at different points of the conversion curve of Fig. 16. A popping one detector together with the median operation makes the conversion curve monotonic again. The arrows indicate the result obtained with no jitter (92-95-94-96, respectively), while the patterned bar shows the median value (92-94-95-96) of the jittered samples. The histograms show that jitter, combined with the median operator, effectively removes the nonmonotonic behavior of the delay line.

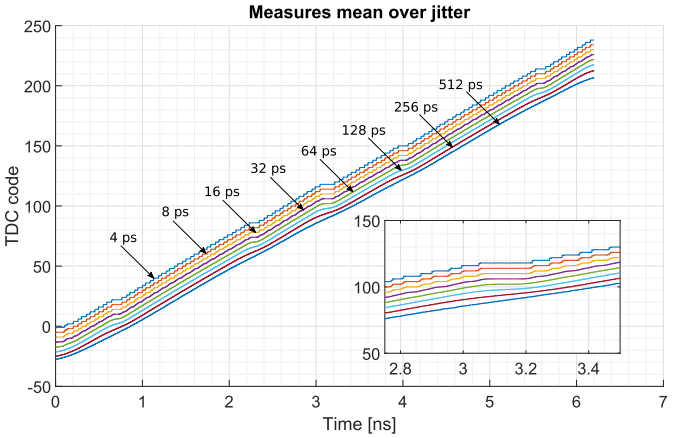

Fig. 18. Simulation results of the jitter linearization effect on the conversion curve by considering the average value for each train of measures.

material effect, its magnitude must be at least of the same order as the LSB. In our simulation, where the data come from the Xilinx physical models, the LSB is 27 ps, since we consider a worst-case scenario with the slowest corner. Consequently, as shown in Fig. 18, a Gaussian Jitter with $\mathrm{FWHM_{jit}}$ of 32 ps is enough to well linearize the characteristic. The effect of jitter is of particular interest when dealing with statistical measures, where a high number of values is collected. On the other hand, we have a degradation of the SSP of the TDC, which varies with the square root of the sum of the squares of the system jitter and the other disturbance sources.

G. Xilinx DCM Performance

We observed that the Xilinx digital clock manager (DCM) that generates the internal frequency of 230 MHz plays an important role in terms of system jitter. In our first setup, a 100-MHz clock was generated by the phase-locked loop of the XEM6001 board. This 100-MHz clock was then fed to the DCM input to generate the 230-MHz output frequency. This process implies a division by 10 and a multiplication

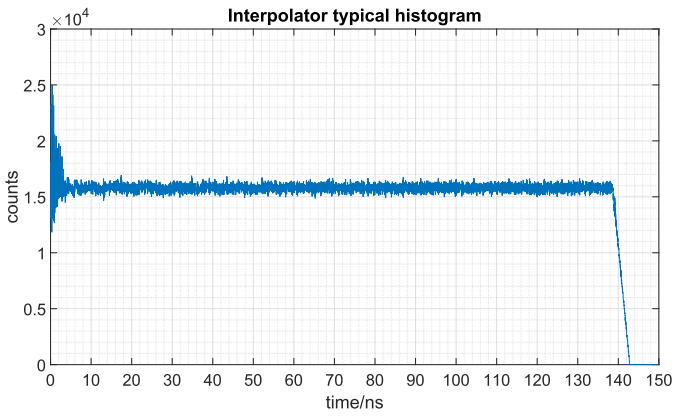

Fig. 19. Interpolating TDC typical density test histogram.

by 23 of the input clock. We then evaluated an alternative solution, by generating a 115-MHz input clock. In this case, the Xilinx DCM simply used a multiplication by 2. To assess the performance of these two approaches, we measured the SSP in terms of LSBs, intended as the standard deviation of a collection of measures of the same delay values averaged over a range of delay values. To do so, we injected multiple delayed pulses from an FPGA clock-synchronous delay source, and recorded the measured values. The minimum and maximum values for SSP improved from $1.43 \div 2.14$ in the $\times 23/10$ case down to $0.69 \div 1.46$ in the $\times 2$ case, which has a simplified clock generation approach.

IV. Experimental Validation

We tested the proposed TDC architecture in a TCSPC laboratory setup and we compared the obtained results with a PicoHarp 300 commercial system in order to assess the reached performance in terms of precision and linearity. TCSPC is a well-established statistical technique, where the fluorescence decay is measured through several excitation–detection cycles, thus overcoming the limitations of a single-shot experiment [23].

The asynchronous interpolating design we implemented improved both differential and integral nonlinearities. Having good linearity figures is fundamental in the field of TCSPC measures, and therefore, enabling curve fitting on TCSPC data to extract the fluorophore lifetime with no calibration. A typical TDC histogram is reported in Fig. 19. DNL and INL have been computed in the $10 \div 135$-ns range, with extreme values of $-0.072 \div 0.070$ and $-0.755 \div 0.872$ LSB, respectively. Resulting DNL and INL is shown in Figs. 20 and 21. A detailed performance summary of our system is reported in Table I. These values compare favorably with the state of the art. Referring to the recent TDC summary table reported by Chen *et al.* [16], our system achieves nonlinearities that are better than designs that use considerably more powerful and modern (and more expensive) FPGAs, such as Virtex and Kintex series, where still peak-to-peak DNLs and INLs bigger than 1 LSB are present. Moreover, we have been able to avoid missing codes (i.e., DNL > -1LSB) starting from the single delay line itself, where other delay line-based TDCs

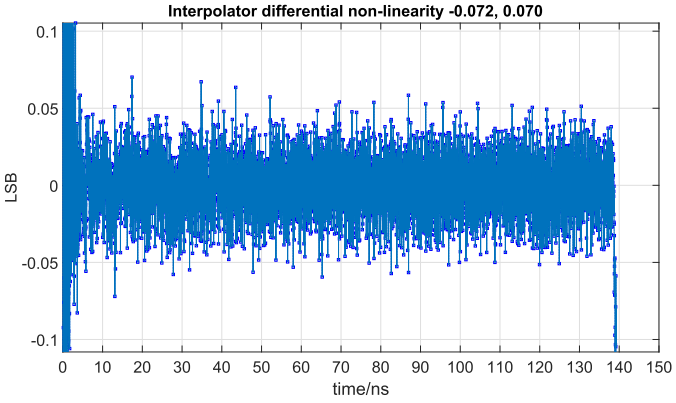

Fig. 20. Interpolating TDC DNL lies in the $-0.07 \div 0.07$ LSB range.

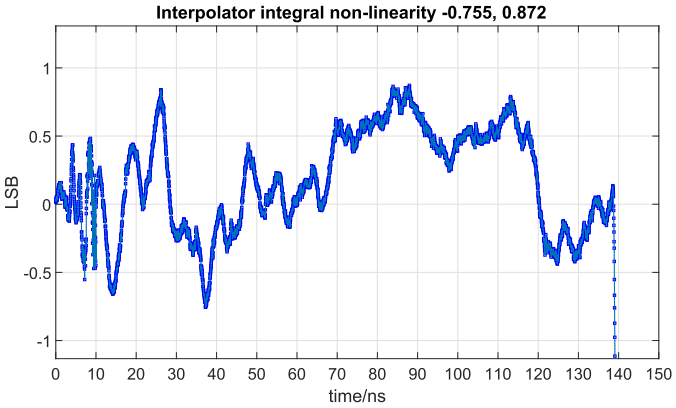

Fig. 21. Interpolating TDC integral nonlinearity lies in the $-0.76 \div 0.87$ LSB range.

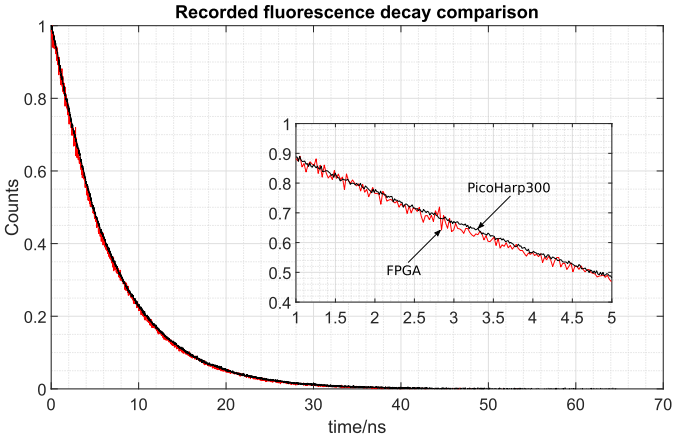

Fig. 22. Fluorescence lifetime decay curve recorded with the proposed FPGA-based system against a commercial PicoHarp300.

lack from this point of view, in some cases with almost half-missing codes [12]. On the other hand, our less advanced FPGA technology results in lower time resolution, together with our decision to promote a better delay line DNL at the expense of the native time resolution. However, even time resolutions in the order of hundreds of picoseconds are acceptable when measuring exponential decays in the range of units of nanoseconds.

A comparison of a fluorescence lifetime decay recorded with our system and the PicoHarp 300 is shown in Fig. 22. The recorded curves have been normalized and superimposed in order to have a clear understanding of the performance of each system. The FPGA-recorded curve still suffers from nonlinearity issues when compared with a state-of-the-art instrument, however, the overall measure quality is sufficiently good to perform an exponential fit to extract the lifetime value. The sample under investigation was a red-emitting Phenyl-Polysiloxane-based scintillator [25], which yielded a measured lifetime of 6.624 and 6.694 ns with the FPGA system and the PicoHarp 300, respectively. The lifetime value, which corresponds to the time constant of the exponential decay, has been automatically computed through data fit with MATLAB. The difference between the two estimations is 70 ps, or only 1.06%, in line with the accuracy specifications of commercial fluorescence lifetime instrumentation [26].

V. Conclusion

Our results show that a single-channel $\simeq$ 26-ps TDC with SSP in the $0.69 \div 1.46$ LSB range can be implemented in a low-cost, low-gate count FPGA. Nonlinearities issues have been addressed acting directly on the delay line structure and conversion techniques, in order to maintain a simpler circuit structure that best suits our target device. A novel study on the statistical code density test methodology has been presented. A two-channel TDC implementing a Nutt interpolation method [19] to extend the measuring range has been implemented as well and tested in a TCSPC setup with good results. The interpolating TDC, due to an asynchronous design that implements the sliding scale technique, reaches DNL and INL values between $-0.057 \div 0.072$ and $-0.705 \div 0.552$ LSBs, respectively, with an SSP in the $1.096 \div 2.815$ LSB range.

The design principles proposed within our work can be used to extend the design to larger and faster FPGAs. Considering the sliding-scale TDC, each channel would take two delay lines and two additional flip-flops for the input stages. A potential application would be to include many channels and perform averaging to reduce the measurement error. In this case, countermeasures are to be taken in order to guarantee uniformity between channels and synchronization of the input signal for each input stage.

References

[1] J. Christiansen, "Picosecond stopwatches: The evolution of time-to-digital converters," *IEEE Solid StateCircuits Mag.*, vol. 4, no. 3, pp. 55–59, 2012.

[2] A. S. Yousif and J. W. Haslett, "A fine resolution TDC architecture for next generation PET imaging," *IEEE Trans. Nucl. Sci.*, vol. 54, no. 5, pp. 1574–1582, Oct. 2007.

[3] P. G. Gutiérrez and A. S. Lleó, "Sistema de medida de tiempo con alta resolución y autocalibrado basado en dispositivo lógico programable," Spain Patent 2 291 057 B2, Feb. 2, 2009.

[4] C. Niclass, M. Soga, H. Matsubara, M. Ogawa, and M. Kagami, "A 0.18-μm CMOS SoC for a 100-m-range 10-frame/s 200$\times$96-pixel time-of-flight depth sensor," *IEEE J. Solid-State Circuits*, vol. 49, no. 1, pp. 315–330, Jan. 2014.

[5] A. Periasamy and R. M. Clegg, *FLIM Microscopy in Biology and Medicine*. Boca Raton, FL, USA: CRC Press, 2009.

[6] P. Keranen, K. Maatta, and J. Kostamovaara, "Wide-range time-to-digital converter with 1-ps single-shot precision," *IEEE Trans. Instrum. Meas.*, vol. 60, no. 9, pp. 3162–3172, Sep. 2011.

[7] C. Veerappan *et al.*, "A 160$\times$128 single-photon image sensor with on-pixel 55 ps 10 b time-to-digital converter," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2011, pp. 312–314.

[8] L. H. C. Braga *et al.*, "A fully digital 8×16 SiPM array for PET applications with per-pixel TDCs and real-time energy output," *IEEE J. Solid-State Circuits*, vol. 49, no. 1, pp. 301–314, Jan. 2014.

[9] J. Kalisz, R. Szplet, J. Pasierbinski, and A. Poniecki, "Field-programmable-gate-array-based time-to-digital converter with 200-ps resolution," *IEEE Trans. Instrum. Meas.*, vol. 46, no. 1, pp. 51–55, Feb. 1997.

[10] A. Balla *et al.*, "The characterization and application of a low resource FPGA-based time to digital converter," *Nucl. Instrum. Methods Phys. Res. A, Accel. Spectrom. Detect. Assoc. Equip.*, vol. 739, pp. 75–82, Mar. 2014.

[11] J. Song, Q. An, and S. Liu, "A high-resolution time-to-digital converter implemented in field-programmable-gate-arrays," *IEEE Trans. Nucl. Sci.*, vol. 53, no. 1, pp. 236–241, Feb. 2006.

[12] C. Favi and E. Charbon, "A 17 ps time-to-digital converter implemented in 65 nm FPGA technology," in *Proc. ACM/SIGDA Int. Symp. Field Programm. Gate Arrays*, 2009, pp. 113–120.

[13] M. W. Fishburn, L. H. Menninga, C. Favi, and E. Charbon, "A 19.6 ps, FPGA-based TDC with multiple channels for open source applications," *IEEE Trans. Nucl. Sci.*, vol. 60, no. 3, pp. 2203–2208, Jun. 2013.

[14] Q. Shen *et al.*, "A 1.7 ps equivalent bin size and 4.2 ps RMS FPGA TDC based on multichain measurements averaging method," *IEEE Trans. Nucl. Sci.*, vol. 62, no. 3, pp. 947–954, Jun. 2015.

[15] N. Dutton *et al.*, "Multiple-event direct to histogram TDC in 65 nm FPGA technology," in *Proc. 10th Conf. Ph.D. Res. Microelectron. Electron.*, Jan. 2014, pp. 1–5.

[16] H. Chen, Y. Zhang, and D. D.-U. Li, "A low nonlinearity, missing-code free time-to-digital converter based on 28-nm FPGAs with embedded bin-width calibrations," *IEEE Trans. Instrum. Meas.*, vol. 66, no. 7, pp. 1912–1921, Jun. 2017.

[17] J. Y. Won and J. S. Lee, "Time-to-digital converter using a tuned-delay line evaluated in 28-, 40-, and 45-nm FPGAs," *IEEE Trans. Instrum. Meas.*, vol. 65, no. 7, pp. 1678–1689, Jul. 2016.

[18] S. Bourdeauducq. (Mar. 2013). "A 26 ps RMS time-to-digital converter core for spartan-6 FPGAs." [Online]. Available: https://arxiv.org/abs/1303.6840

[19] J. Kalisz, "Review of methods for time interval measurements with picosecond resolution," *Metrologia*, vol. 41, no. 1, p. 17, 2003.

[20] C. Cottini, E. Gatti, and V. Svelto, "A new method for analog to digital conversion," *Nucl. Instr. Meth.*, vol. 24, pp. 241–242, Aug. 1963.

[21] B. Markovic, S. Tisa, F. A. Villa, A. Tosi, and F. Zappa, "A high-linearity, 17 ps precision time-to-digital converter based on a single-stage Vernier delay loop fine interpolation," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 60, no. 3, pp. 557–569, Mar. 2013.

[22] J. Y. Won, S. I. Kwon, H. S. Yoon, G. B. Ko, J.-W. Son, and J. S. Lee, "Dual-phase tapped-delay-line time-to-digital converter with on-the-fly calibration implemented in 40 nm FPGA," *IEEE Trans. Biomed. Circuits Syst.*, vol. 10, no. 1, pp. 231–242, Feb. 2016.

[23] W. Becker, *Advanced Time-Correlated Single Photon Counting Techniques*, vol. 81. Berlin, Germany: Springer, 2005.

[24] O. Kelly. (Mar. 2015). *Front Panel, User Manual*. [Online]. Available: http://assets00.opalkelly.com/library/FrontPanel-UM.pdf

[25] M. D. Palma *et al.*, "Red emitting phenyl-polysiloxane based scintillators for neutron detection," *IEEE Trans. Nucl. Sci.*, vol. 61, no. 4, pp. 2052–2058, Aug. 2014.

[26] K. J. Petersen, K. C. Peterson, J. M. Muretta, S. E. Higgins, G. D. Gillispie, and D. D. Thomas, "Fluorescence lifetime plate reader: Resolution and precision meet high-throughput," *Rev. Sci. Instrum.*, vol. 85, no. 11, p. 113101, 2014.