Measurement 238 (2024) 115313

Contents lists available at ScienceDirect

Measurement

journal homepage: www.elsevier.com/locate/measurement

Novel machine learning-driven optimizing decoding solutions for FPGA-based time-to-digital converters

Fabio Garzetti, Nicola Lusardi *, Enrico Ronconi, Andrea Costa, Angelo Geraci

DEIB, Politecnico di Milano, 20133 Milano, Italy

ARTICLE INFO

Keywords:
Bubble errors
Calibration
Tapped delay-line time-to-digital converter (TDL-TDC)
Field programmable gate array (FPGA)
Neural network

ABSTRACT

Calibration of models and data structures is recurring in a large number of cross-cutting applications from finance to engineering. Even though there are numerous and well-established specific calibration techniques for each application sector, using Neural Networks (NNs) can improve performance. For instance, Tapped Delay-Line Time-to-Digital Converters (TDL-TDCs) implemented in Field Programmable Gate Arrays (FPGAs) are increasingly being used in a variety of research applications, such in the time-resolved spectroscopy or in medical imaging mainly for their high-precision and flexibility. Specific decoding on the sampled information from the TDL, together with calibration to compensate for non-idealities, (i.e., Bubble Errors, BEs, and Process–Voltage–Temperature fluctuations, PVTs) are carried out for generating the conversion of digital codes to time units. In this fundamental process, the impact of Machine Learning (ML) usage has not yet been investigated. In this paper, focusing on advanced FPGA devices (i.e., 28-nm, 20-nm, and 16-nm), we propose an approach based on NNs running in Python on a standalone PC to identify the optimal conversion from digital codes to timestamps, comparing it with the classical fully FPGA-based solution c literature. The experimental validations are performed on Artix-7 (XC7A100TFG256-2) and Kintex UltraScale (XCKU040-FFVA1156-2-E) in 28-nm and 20-nm technology nodes, achieving precision of 12.9 ps r.m.s. and 4.85 ps r.m.s., respectively. These results are in line with the state-of-the-art, demonstrating that in 28-nm technology, the bubble compression algorithm is sufficient to achieve high-precision, while reordering mechanism is crucial to compensate for BEs within the 16/20-nm technology node.

1. Introduction

In many industrial and research fields, such as laser range-finders [1], time-mode Analog-to-Digital Converters (ADCs) [2], time-resolved spectroscopy [3–6], Phase-Locked Loops (PLLs) [7], Time-Correlated Single Photon Counting (TCSPC) [8,9], and Time-of-Flight Positron Emission Tomography (TOF-PET) [10–12] just to name a few, Time-Interval-Meters (TIMs) are key elements of implementations. Numerous designs of TIMs have been developed using various technologies [13]. The Time-to-Amplitude Converter (TAC) [14], the first TIM proposed historically, converts the time interval under measurement in a voltage level by storing charge in a capacitor for the interval duration. In comparison to analog and mixed-signal TIMs, fully-digital TIMs like Time-to-Digital Converters (TDCs) have gained favor in the digital age. Application Specific Integrated Circuits (ASICs) or Programmable Logic Devices (PLDs), including Field Programmable Gate Arrays (FPGAs) and System-on-Chip (SoC), can be utilized to implement TDCs [15,16]. FPGAs offer the fastest prototyping and most efficient research solutions [17,18], providing huge flexibility, low Non-Recurring Engineering (NRE) costs, and performance on par with cutting-edge ASIC-based TDCs [19].

Various FPGA-based architectures are present in the literature, with the most common ones being the Tapped Delay-Line TDC (TDL-TDC) [16], Vernier Delay-Line TDC [20], Ring-Oscillator TDC [21,22], Gray Counter Oscillator TDC (GCO-TDC) [23], Multi-Phase and Single-Phase Shift Clock Fast Counter TDC (SCFC-TDC) [24], just to mention a few of the main ones. Each of these architectures matches at best to a set of resolution, conversion rate, and area specifications. For instance, SCFC-TDC and GCO-TDC are small area options with high conversion rates but limited resolution while Ring-Oscillator TDC (RO-TDC) offers high-precision within low area occupancy but an extremely high dead-time. Vernier Delay-Line TDC (VDL-TDC), on the other hand, uses more space but offers a greater resolution at slower conversion rates. The TDL-TDC is the most balanced and practical choice when looking for the optimum compromise between resolution, area, energy

* Corresponding author.
E-mail addresses: fabio.garzetti@polimi.it (F. Garzetti), nicola.lusardi@polimi.it (N. Lusardi).

https://doi.org/10.1016/j.measurement.2024.115313
Received 4 December 2023; Received in revised form 3 July 2024; Accepted 12 July 2024
Available online 24 July 2024

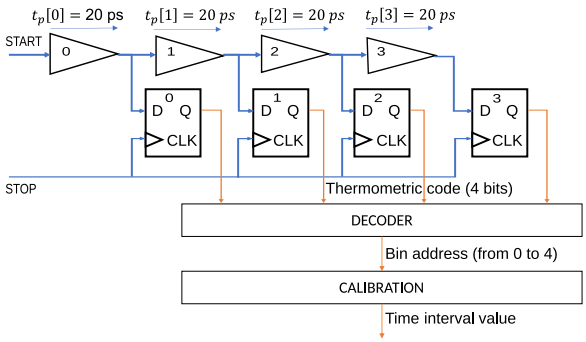


Fig. 1. Schematic structure of the TDL-TDC with $N = 4$ taps with propagation delay $t_p[i] = 20$ ps, $i \in [0; 3]$.

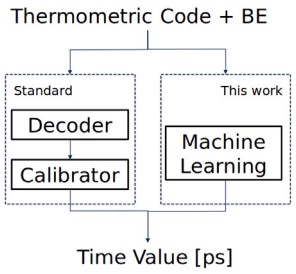


Fig. 2. Comparison between the standard approach of decoding and calibration sequence versus the proposed ML-based approach.

consumption, and conversion rate. This is the reason a TDL-TDC was chosen to contextualize the novel decoding method that is proposed.

In TDL-TDCs, a digital low-to-high step signal (referred as START) goes through a chain of $i \in [0; N-1]$ buffers (known as taps or bins) with propagation delay $t_p[i]$, which stores the high value of the propagating step in D-type Flip-Flops (DFFs), one for each buffer and initialized to low logic level. The values at the input of DFFs are sampled when another digital low-to-high step signal (referred to as STOP) occurs. In this way, the length of time interval beginning at START edge and ending at STOP one (Fig. 1) is measured as a series of consecutive high-logic values, known as thermometric code. Least Significant Bit (LSB), which represents the resolution of the measurement, is fixed by the propagation delay of each buffer. This output of the chain of buffers, called Tapped Delay-Line (TDL), is transformed into the final time value, known as a timestamp.

The primary criticalities in this processing flow, when taking into account the FPGA implementation, are the Bubble Errors (BEs) [25], a switching effect in the thermometric code, and the value dispersion of the tap propagation delays [26,27]. Regarding BEs, DFFs and their connections have a variety of non-idealities and mismatches, which could lead to some inconsistencies in the output rather than being a continuous string of high levels typical of a correct thermometric code [28,29]. These errors, known as BEs, manifest as one or more zeros appearing as interruptions (i.e., like bubbles) in the continuity of ones. For example, the output might be "11111010" instead of "11111110" (where the position of 1 and 0 represent the propagation over the TDL from left to right as represented in Fig. 1). These BEs may be due to stochastic processes (e.g., sampling errors caused by violations of setup and hold times) or actual deterministic non-linearities present in TDL propagation. A standard thermometric-to-binary converter (a.k.a., decoder or in some papers also called encoder) will produce an inaccurate value (i.e., the result will be close to the expected, only that shifted by the number of bits on which the bubble error occurred) when fed this faulty sequence since it expects to receive only a thermometric code as input. Therefore, when BEs are present, actual decoders are employed to convert the output "pseudo thermometric code" containing BEs into a code with a sharp change from ones to zeros. As a result, there is a merging of TDL bins (e.g., "11111010" is interpreted as "11111100" or "11111110") with the possibility that more (pseudo) thermometric codes (where pseudo in brackets means that the BEs could or not be preset) correspond to a single timestamp increasing the quantization error [30]. So, the thermometer code represents the propagation order within a TDL. In case of BE, we have a pseudo-thermometer code, and therefore, we are unable to define the propagation order a priori, whether 11111010 is actually a 11111100 (6 taps propagated) or 11111110 (7 taps propagated). By removing the BEs, it is possible to perform the measurement without increasing the quantization error [31]. Furthermore, since Process–Voltage–Temperature (PVT) fluctuations can cause variations in the nominal features of the taps, a calibration procedure is required to account each tap's actual propagation delay [32]. This results in the estimation of a characteristic curve that links each decoded (pseudo) thermometric code into the corresponding real time value.

Therefore the process of synthesis of the time information involves two phases, the first of decoding and the second of calibration working as a bijective function that maps one and only one (pseudo) thermometric code to one and only one final timestamp in time units (e.g., picoseconds).

The aim of this work, never been previously explored in the literature, is to implement the conversion function (i.e., from digital codes to timestamps) in Python on a standalone PC with the presence of BEs using a Machine Learning (ML) approach [33]. The ML approach is simply the tool employed. Due to the absence of prior experiments, the NN was constructed based on the hardware structure of the decoders and calibrators, optimizing it with a standard Adaptive Moment Estimation algorithm.Our goal is to minimize the quantization error by addressing BEs and compare the results with fully FPGA-based solutions available in the scientific literature (Fig. 2). The ML output can function as a verification element, aiding in the selection of decoding solutions from the various options available in the scientific literature. Moreover, it is been highlighted the presence of the "bin merging" phenomena that negatively impact on the TDL-TDC's precision.

Main decoding and calibration standards addressing non-idealities affecting the TDL of TDL-TDC are introduced in Section 2. Section 3 describes how the ML approach has been applied to the TDC problem. Finally, experimental validations on modern technology nodes, the 28-nm high-k metal gate (HKMG) and the 16/20-nm Fin Field-Effect Transistor (FinFET), are carried out on 28-nm 7-Series Artix-7 (XC7A100TFG256-2) and 20-nm Kintex UltraScale (XCKU040-FFVA1156-2-E) in Sections 4 and 5. The results are consistent with the state-of-the-art, with a precision of 12.9 ps r.m.s. and 4.85 ps r.m.s. for the 28-nm and 20-nm technological nodes, respectively. Moreover, thanks to the ML approach, it was demonstrated that in the 28-nm technology node, the bubble compression algorithm alone is adequate to achieve high precision, whereas in the 16/20-nm technology node, a reordering mechanism becomes essential to mitigate BEs.

2. Decoding and calibration procedures

We start from the ideal TDL (described in Section 2.1) to get to its realization (discussed in Section 2.2), because the decoding issues are dependent on the non-linearities that come along with the TDL implementation.

2.1. Ideal Tapped Delay-Line

In the ideal TDL case, there are no BEs and all the propagation times of the buffers that make up the line are equal (i.e., t_p). The system needs a decoder just to compress the thermometric information to pure binary. In particular, the length in time T_{meas} of the interval

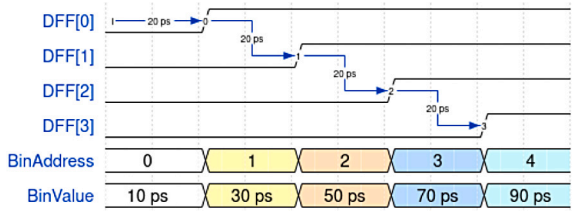


Fig. 3. Example of 90 ps-long interval measured by a TDL-TDC with ideal 4-tap TDL. DFFs are the flip-flop outputs.

under measurement is obtained by multiplying the number $BinAddress$ of buffers, whose output is high (i.e., the thermometric code decoded into binary one), by the ideally constant propagation delay t_p. The propagation delay t_p is also the LSB of the measurement. The map converting digital information in time units is called $BinValue[i]$, where $i \in [0; N-1]$ is the $BinAddress$ and N is the total number of buffers constituting the TDL. Just for having T_{meas} zero average value and, consequently, $LSB/\sqrt{12}$ as standard deviation (i.e., precision) [34,35], the $BinValue$, the $BinAddress$ (i.e., i in equations) and t_p are related as

$$\begin{cases} BinValue[0] = \frac{t_p}{2} & i = 0 \\ BinValue[i] = BinValue[i-1] + \frac{t_p}{2} & i \in [1; N-1] \end{cases} \quad (1)$$

So, if the TDL is ideal, the decoder is simply a thermometric-to-binary converter that counts the number of ones in the TDL, compressing the information from N to $log_2(N)$ bits. Fig. 3 shows the schematic flow, through timing diagrams, of the measurement system that consists of 4-tap TDL, decoder (from thermometer-code to $BinAddress$) and calibration stage (from $BinAddress$ to $BinValue$).

This basic version of the TDL-TDC is usually accompanied by Nutt-Interpolation to extend the Full-Scale Range (FSR) of the measurement without making the number of buffers of the TDL diverge [36]. This consists in splitting the measurement T_{meas} in a coarse part performed by a coarse counter (i.e., T_{coarse}), and a fine part (i.e., $T_{1,fine}$ and $T_{2,fine}$) calculated by the TDL-TDC. The coarse counter counts the number ΔN of whole periods T_{CLK} contained between START and STOP edges (i.e., $T_{coarse} = \Delta N \cdot T_{CLK}$) [37]. The TDL-TDC evaluates the distances $T_{1,fine}$ and $T_{2,fine}$ of the START and STOP instants respectively from the edges of the clock signal closest to them.

$$T_{meas} = T_{coarse} - T_{2,fine} + T_{1,fine} \quad (2)$$

The larger the dimension of the coarse counter, the more extended the FSR is.

2.2. Real Tapped Delay-Line

Due to the physical implementation, the TDL in an FPGA device experiences the effect of BEs [25] and PVT fluctuations [38]. Let us look at the resources of the devices utilized to realize the TDLs to see where these issues originate. As native resources with configurable TDL are not available in FPGA devices [15], these must be implemented as structures made up of elements having different purposes such as logic blocks [39], routing arrays [40], Digital Signal Processing (DSP) blocks [41], and belonging to the FPGA fabric. Among these choices, the carry propagation resources (a.k.a., CARRY[1]), which in Xilinx technology are the CARRY4 (28-nm 7-Series) and the CARRY8 (20-nm UltraScale and 16-nm UltraScale+), are the most suitable given the abundant availability in the devices as parts of the Configurable Logic Blocks (CLB) [43,44]. Among other things, they also prove to be the best compromise between linearity, speed and uniformity of propagation delay values, $t_p[i]$. The simplified illustration of the CLB in Fig. 4 puts in evidence the CARRY blocks that implement the taps (MUX with Sel at '1') and DFFs, which are the TDL's building blocks.

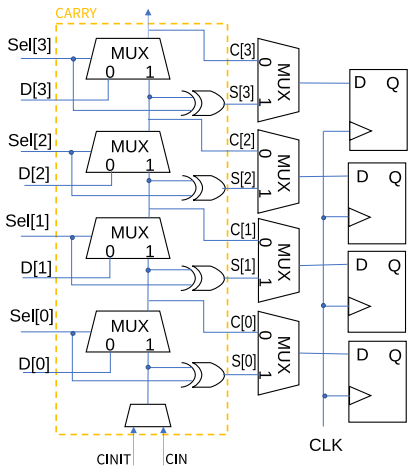


Fig. 4. Simplified representation of a CLB that highlights the key components involved in the TDL implementation.

One issue emerges from the fact that these CARRY stages have not the necessity of propagation delays that are equal and stable with regard to PVT fluctuations [26] for the function they must perform. The non-uniformity of the individual delays results in the non-linearity of the propagation time, which consequently needs a real-time "bin-by-bin" calibration in order to know at all times during the operation the better estimation, called $BinWidth[i]$, of the real propagation delay $t_p[i]$ of the ith buffer (Section 2.2.1).

Instead, the BEs are derived from the DFFs' hardware limitations and the CARRY blocks' variations in layout and routing (Section 2.2.2). As evidence of how important the dispersion of propagation delay values and BEs are, also in relation to the chosen device, Table 1 reports the mean value (a.k.a., mean-bin) and the maximum value (a.k.a., ultra-bin) of the $t_p[i]$, and the length/depth of the BE (a.k.a., Max Bubble Depth [45], MBD) measured in TDLs implemented in various AMD/Xilinx and Intel/ALTERA FPGAs.

2.2.1. Delay values dispersion and calibration

Given the dispersion of delay values that cannot be eliminated, it is necessary to best estimate the real values of each one of them to be used in calculating the timestamp. For doing this bin-by-bin calibration, we resorted to the Code Density Test (CDT) technique [61] to estimate for each ith buffer the $BinWidth[i]$ value to be used in (1) in place of the nominal t_p, that is

$$\begin{cases} BinValue[0] = \frac{BinWidth[0]}{2} & i = 0 \\ BinValue[i] = BinValue[i-1] + \frac{BinWidth[i]}{2} & i \in [1; N-1] \end{cases} \quad (3)$$

[1] The nomenclature CARRY is to be referenced to the AMD/Xilinx vendor upon which the experimental data in the manuscript is based, while for Intel/Altera devices, different nomenclature is utilized. Carry-chains in Intel/Altera FPGAs are built by connecting in series the carry logic of Logic Array Blocks (LABs). In Cyclone I, II, and IV, each LAB contains either 10 (in the case of Cyclone I) or 16 (in the case of Cyclone II and IV) Logic Elements (LEs), each of which has one tap of the carry chain, resulting in 10 or 16 taps per LAB respectively. Additionally, in Cyclone V and Arria 10, each LAB contains 10 Adaptive Logic Modules (ALMs), each of which has two elements of the carry chain, resulting in 20 taps per LAB. "Furthermore, in modern FPGAs (Intel/Altera series V and 10, or Xilinx UltraScale and UltraScale+), each tap of the carry chain can be connected to two flip-flops, allowing the dual-sampling technique [42]. However, this technique is not taken into account in this contribution.

Table 1
Comparison of TDLs implemented in various AMD/Xilinx and Intel/ALTERA FPGAs in terms on non-idealities (all the information listed in the table is available in the documents cited in the last column).

FPGA	Vendor	Tech. Node	Bin		MBD	Tap per CARRY or LAB	Ref
			Mean	Max			
		[nm]	[ps]	[ps]	[–]	[–]	
Cyclone I	Intel/Altera	130	70	170	0	10	[46]
Cyclone II	Intel/Altera	90	45	155	0	16	[47]
Cyclone-IV	Intel/Altera	60	58	86	2	16	[48]
Virtex-4	AMD/Xilinx	90	45	100	1	2	[49]
Virtex-5	AMD/Xilinx	65	34	110	2	4	[50]
Cyclone V	Intel/Altera	28	5.98	32.2	3	20	[51]
Spartan-6	AMD/Xilinx	45	25.57	75.76	>1	4	[52]
Virtex-6	AMD/Xilinx	40	10	20.9	2	4	[53]
Artix-7	AMD/Xilinx	28	22.2	47	4	4	[54]
Kintex-7	AMD/Xilinx	28	10.35	25	2	4	[55]
Virtex-7	AMD/Xilinx	28	23	40	4	4	[56]
Cyclone 10	Intel/Altera	20	10	80	High	20	[57]
Arria 10	Intel/Altera	20	9.1	70	High	20	[58]
Kintex UltraScale	AMD/Xilinx	20	4.18	60	16	8	[59]
Zynq UltaScale+	AMD/Xilinx	16	1.9	23.18	16	8	[60]

The CDT entails taking a sufficiently large K number of measurements of intervals with uniformly distributed random lengths between 0 and T_{CLK}, where T_{CLK} is the basic TDL-TDC's FSR under Nutt-Interpolation, or maximum interval length. The K measurements are stored in a histogram (i.e., $CDT_{hist}[i]$) that bined the range $0 - T_{CLK}$ into the $i \in [0; N-1]$ taps of the TDL. From the $CDT_{hist}[i]$ histogram, the ith bin width results to be $binWidth[i] = \frac{CDT_{hist}[i]}{K} \cdot T_{CLK}$. The thus calculated bin widths are saved in a Look-Up Table (LUT) named Calibration Table (CT). It can be demonstrated that the δt_{CAL} error between the real $t_p[i]$ and the statistically estimated $binWidth[i]$ is on the order of T_{CLK}/K. As a result, if K is sufficiently large, the error can be ignored [62,63].

When considering working under calibrated conditions, from the value of each $BinWidth[i]$ (with $i \in [0; N-1]$), it becomes possible to estimate the so-called Equivalent LSB (a.k.a., LSB_{EQ}) (4), from which it is possible to extract the quantization error (i.e.; $\sigma_Q = LSB_{EQ}/\sqrt{12}$) [34,64].

$$LSB_{EQ} = \sqrt{\frac{1}{\sum_{i=0}^{i=N-1} BinWidth[i]} \cdot \sum_{i=0}^{i=N-1} BinWidth^3[i]} \tag{4}$$

2.2.2. Bubble errors and decoding

Discriminating between deterministic and stochastic causes of BEs is necessary.

About deterministic factors, the former one is related to the routing of connections between buffers and DFFs with different propagation delays and to the skew associated with the clock signal that synchronizes the DFFs, whose effects, at least theoretically, might be corrected. As seen in Fig. 5, the signal experiences a delay as it travels from the output of each CARRY tap to the corresponding DFF's input and from one CARRY block to the next one [65]. Additionally, the clock skew, which is the arrival of the clock at various components at different times, is a major cause of the issue [27]. The clock signal is distributed hierarchically throughout the FPGA to reduce the skew, beginning with the clock regions in which the device is divided. To ensure a very low skew (i.e., below picosecond), the clock is routed to all CLBs before being distributed to the DFF. However, skews are not negligible when different clock regions are used by the implementation (i.e., tens of picoseconds). Fig. 6 depicts possible effects of the presence of the clock skew on the generation of the thermometric code. The architecture of FPGAs does not minimize routing concerns because they solely impact asynchronous signals. Simply put, they alter each buffer's apparent delay, which raises delays dispersion.

If the delay introduced by skew and routing is small compared to the propagation delay of the buffers $t_p[i]$, no matter occurs, in particular no BEs. Differently, if the propagation delay of the buffers, $t_p[i]$, is

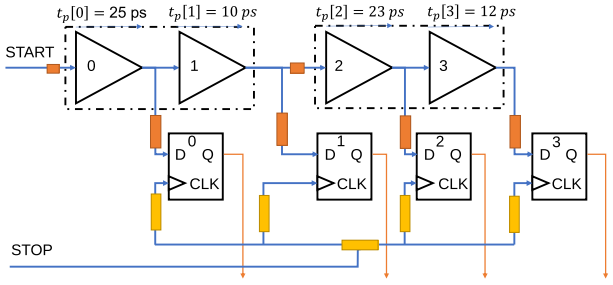


Fig. 5. Real 4-tap long TDL-TDC architecture. The orange rectangles represent extra delays due to routing of the step signal, while the yellow rectangles represent the clock skew due to the distribution network. The dotted rectangles covering the buffers represent the CARRY blocks with 2 taps per block.

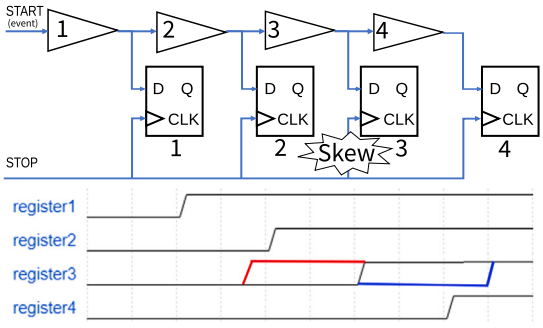


Fig. 6. Example of the clock skew's effect on a 4-tap TDL where only the third bin suffers from clock skew (top). The effect of positive skew is indicated in red, while the effect of negative skew is indicated in blue (bottom) causing BEs.

shorter than the delay caused by skew and routing, the switching order of the corresponding outputs may vary, introducing the BEs in the thermometric code and making decoders fail. The two decoding techniques known as "sum1s" [31] (a.k.a., bubble compression), which counts the number of 1s in the code string, and "Log2" [62] (a.k.a., one-hot),[2] which looks for the transition from 1 to 0 in the code

[2] The names Log2 and one-hot have been derived from this type of decoder because this decoding scheme resembles the operation of base 2 logarithm and the one-hot code; over the TDL only (pseudo) thermometric codes are propagated.

string,[3] are the two that are most frequently used in the literature when dealing with FPGA-based implementation. So, for instance, referring to Fig. 5, ideally the four measurable lengths (i.e., STOP occurring when START reaches buffer 1, 2, 3 or 4) correspond to thermometric codes "1000", "1100", "1110" or "1111" respectively. Both decoding algorithms work properly. In presence of BEs, for the same interval lengths the (pseudo) thermometric codes could be "0010", "1010", "1011" or "1111" respectively. If this were the case, algorithm sum1s would work correctly (i.e, "1", "2", "3", and "4") while algorithm Log2 would not (i.e., "0010" and "1010" are decoded as "3", while "1011" and "1111" as "4"). If instead the generated (pseudo) thermometric codes were "1000", "1100", "0110" or "0011" respectively, only algorithm Log2 (i.e., "1", "2", "3", and "4") would works while sum1s fails (i.e, "1100", "0110", and "0011" are decoded as "2"). If neither of the two algorithms could be applied without faults, it would be necessary for the decoder to also swap reordering the content of the code strings to return to one of the favorable cases. This could be extremely complex but feasible as the effects of skew and routing are deterministic and therefore correctable [30].

However, in addition to deterministic skew and routing errors, there are statistical errors due to metastability phenomena that can generate BEs but cannot be predicted and consequently corrected. Indeed, since the TDL is asynchronous, if the sampling of the buffer output violates the timing parameters (i.e., setup and hold times) of the DFFs, these can enter a metastable state [28] and therefore, as is known, resolve the respective outputs randomly as 0 or 1, thus being able to introduce BEs in the thermometric sequence. Considering a 2-tap TDL without skew and routing issues but in presence of random setup and/or hold time violations, one different pseudo thermometric code, "01", might be statistically generated between the deterministic sequences (with no timing violation) "00", "10", and "11", which only depends on the START-STOP distance with no timing violation. Referring to Fig. 7, the ideal case corresponds to deterministic codes "00", "10", or "11" with 100% probability. In presence of random timing violations, the DFFs' outputs Q[0] and Q[1] can be either "0" or "1", resulting in different codes with different probabilities depending on how metastability is resolved. Due to this, the introduced error is statistical and, unlike the deterministic BE caused by skew and routing issues, cannot be corrected a priori. Returning to the 4-tap TDL and to the possible (pseudo) thermometric codes "0000", "0010", "1010", "1011" or "1111" that in presence of deterministic BEs can be decoded by the sum1s algorithm, adding DFFs timing violations could generate with a certain probability, for instance, the sequence "1000", "1010", "1010", "1011", "1111", with which the sum1s decoding would no longer work. So, the main problem with BEs presence is essentially the increase in the number of measurable pseudo thermometric sequences, whose occurrence is statistical.

Fig. 8 shows sum1s and Log2 decoding approaches adopted on the real TDL of Fig. 5 in presence of both routing and skew issues and setup and hold time violations. Both methods merge bins reduced from 7 (pseudo) thermometric codes (i.e., "0000", "0010", "1010", "1000", "1100", "1110", and "1111") to 5 (i.e., "0", "1", "2", "3", and "4"), increasing the quantization error of the TDL-TDC. It can also be seen that the Log2 algorithm may produce a zero width bin (i.e., "2"), due to the change of Q[2]'s order in time with respect to the physical order in the TDL due to the skew and routing issues. To solve this, a block that reorders the bits of the pseudo thermometric code should be positioned between the TDL output and the decoder input [30,66,67],

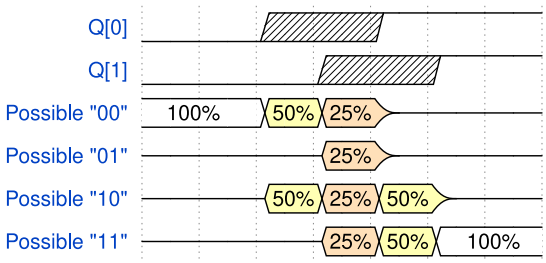


Fig. 7. Waveforms showing the statistical effect on the generated (pseudo) thermometric code due to timing violations in the couple of DFFs constituting a 2-tap TDL.

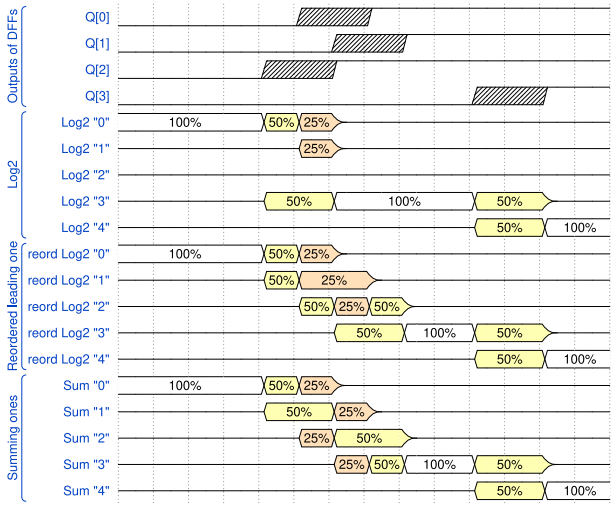
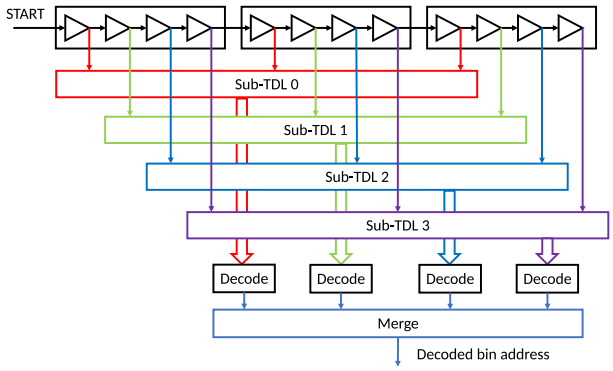


Fig. 8. The timing diagram shows all cases on a 4-tap TDL with the decoding algorithms Log2 (bin address is the position of the first 1) and sum1s (bin address is the sum of the ones), which are respectively the one that ignores the BE (Log2) and the one that considers all BE (sum1s). Furthermore, it also represents the "reordered leading one" [30] (a.k.a., reord Log2), that is the reordering of the taps to compensate deterministic BEs and successive Log2 decoding. The output of decoders is represent as decimal number in ordinate.

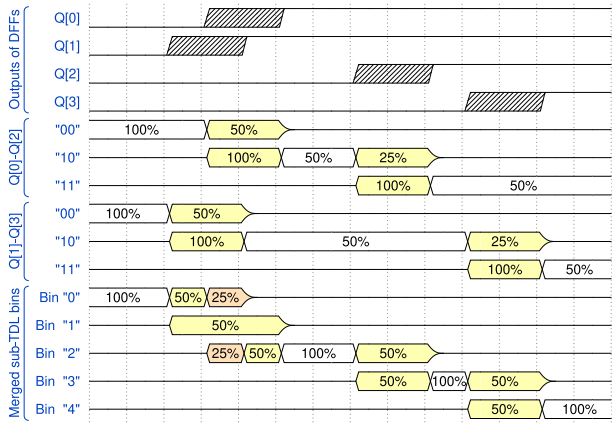
giving rise to the "reordered leading one" algorithm (i.e., reordering of bins and decoding with Log2). To decide the reordering, someone uses tools to simulate the clock skew in the TDL implementation (meaning that the reordering has to be done by hand), while others iteratively reorder the pseudo thermometric code's bits until no zero width bins appear. Instead, the sum1s algorithm is intrinsically quite resistant to the disordering caused by the skew and routing issues, and does not need additional reordering.

Other decoding techniques exist, like merging sub-TDL [59,68] (Fig. 9). Let us suppose that the sources of BE occur only inside a MBD. In this case, the TDL can be divided into smaller sub-TDLs free of bubble errors (each tap of each MBD forms part of a sub-TDL, as shown in Fig. 9), which can be decoded as if they were ideal TDLs. Separating the TDL into sub-TDLs, determines greater quantization error that can be reduced if the sub-TDLs are merged. The most common way to merge the decoded sub-TDLs is using a simpler adder in a similar way as sub-Interpolated multi-chain TDL-TDC in Super Wave Union [69]. The decoding obtained from the sub-TDL algorithm will be identical to the sum1s (i.e., also in sub-TDL, an output equal to the number of '1's present in the pseudo thermometric code is produced), therefore, they are two different hardware circuits that produce the same result. It follows that the choice between sub-TDL and sum1s decoder is purely implementation-based, meaning that the one that best fits in terms of area, timing, and power dissipation is used for the architecture of the FPGA being utilized. For this reason, we can assume that the output

[3] As can be seen in Fig. 5, and as is generally represented in the scientific literature, the propagation occurs from left to right. Usually (as has been done in this paper), this convention is maintained in (pseudo) thermometric code strings as well. Consequently, the least significant bit is on the left while the most significant bit is on the right (e.g., "1000" represents the decimal number 1, while "0001" represents the decimal number 8).



(a) Sub-TDL structure in a TDL characterized by $MBD = 4$.



(b) Waveforms of merged sub-TDLs decoding in a TDL composed by $MBD = 2$.

Fig. 9. Example of merged sub-TDLs decoding.

obtained with the sub-TDL decoder is consistent with that of the sum1s decoder; thus, for our purposes, they are equivalent solutions.

The conclusion is that in presence of deterministic and stochastic causes it is unfeasible to identify the best decoding mechanism to solve the BEs. Moreover, timing violations increase the entropy of the system, making its solution even more challenging. These considerations open the way to the ML approach.

3. Machine learning approach

In Section 3.1, an overview on what concerns with ML and Neural Networks (NNs) preparatory to the following discussion is carried out. Section 3.2 summarize the adopted NN; then, in Section 3.3 a novel thermometric-to-time training procedure addressed to Nutt-Interpolated TDL-TDCs based on supervised learning is proposed. All the algorithms developed in Sections 3.2 and 3.3 have been implemented in Python.

3.1. Overview

Machine Learning (ML) [33] generates algorithms from observation of experimental data. The principal learning methods are the supervised learning, unsupervised learning and reinforcement learning [70]. Supervised learning uses a set of samples and the corresponding desired output, a.k.a label, to approximate a function that generalizes for unknown inputs. Compared to unsupervised learning and reinforcement approaches, supervised learning is simpler and more straightforward when you have a dataset where both the input and output are known [71]. Supervised learning can be divided into classification and regression problems [72]. Classification problems map an input to one

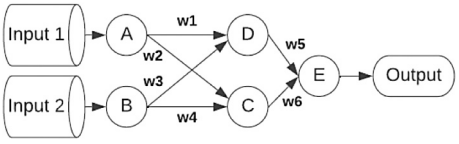


Fig. 10. Example of a NN. Nodes A and B form part of the input layer, node E forms part of the output layer, and nodes C and D form part of the hidden layer. A NN must always have one input and output layers, but may have any number of hidden layers.

category from a set of k classes, producing a function $f(\cdot)$ that maps $\mathbb{R}^n \rightarrow [1, \ldots, k]$. An example is to decide if an image depicts a cat, a dog, or neither. In regression problems, a numerical value is predicted from an input, producing a function $f(\cdot)$ that maps$\mathbb{R}^n \rightarrow \mathbb{R}$. An example is to predict the future value by using preceding data. The problem at hand is clearly a regression, as a numerical value of time must be predicted from a (pseudo) thermometric code. For regression, due to the complexity of this problem a NN approach has been selected [73].

The NNs [33] are complex functions that consist of a network of interconnected nodes, as Fig. 10 shows. Each node has a set of inputs, weights, and an activation function and its output is defined by $y = f(x \cdot \theta)$, where x is an array of the node's inputs $x = [x_1, x_2, \ldots, x_n]$, θ is an array of each input's weight $\theta = [\theta_1, \theta_2, \ldots, \theta_n]^T$, and $f(\cdot)$ is the activation function; e.g., Rectified Linear Unit (ReLU), Leaky ReLU, Sigmoid [74]. The NN's behavior depends on the number of layers, number of nodes on each layer, class of activation functions and values of the weights. The values of the weights can be chosen through the learning process (e.g., supervised, unsupervised, reinforced) with a proper strategy. Back-propagation is the most common one used in supervised learning consists of a series of forward and backward steps through the NN. The forward step presents an input to the NN and waits for the output predicted by the NN. Afterwards, each layer is trained through supervised learning, starting with the last layer and finishing with the first one (Fig. 11). The predicted output is compared with the label through a loss function; e.g., Mean Squared Error (MSE), Mean Absolute Error (MAE), Huber Loss. The weights are initialized using a proper initialization algorithm (e.g., He, Xavier) and then modified during the supervised leaning by using an optimization algorithm, e.g., Adaptive Moment Estimation (ADAM), Nesterov-accelerated Adaptive Moment Estimation (NADAM), Root Mean Square Propagation (RMSprop), usually based on minimizing or maximizing the loss function through the partial derivative of the loss function with respect to each weight [75]. Once the weights of the last layer are changed, the label which is at the output is passed backwards to the input of the last layer and the penultimate layer is then trained with the modified label. The procedure is repeated until all layers are trained. NNs have lots of settings, a.k.a hyperparameters [33], to be selected: number of layers, number of nodes per layer, number of connections between the nodes of two layers, activation functions of the nodes, choice of optimization algorithm, number of passes through the training data, etc. To select the hyperparameters, cross-validation is usually used. The training data is split in two sets, a training set and a validation set; the neural network is trained with the training set, and then the trained NN is used to predict the results of both the training and validation sets. If the NN has not enough parameters to represent the chosen function, then the loss function of both sets will be poor. This effect is referred as underfitting in literature [33]. Conversely, if the NN fits data too well to the training set, losing the generalization property, the loss function on the validation set is much worse than the loss function on the training set. This effect is referred as overfitting in literature [33]. The NN hyperparameters are selected by iterative trimming and observing the results on the training and validation sets, although a final test with previously unseen data needs to be done to check the NN's usefulness.

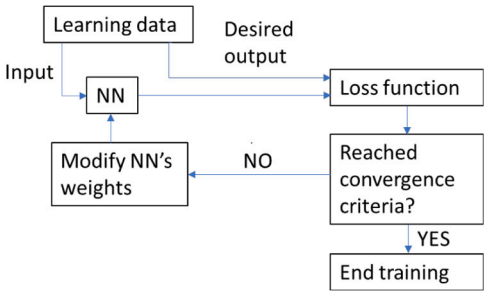


Fig. 11. Back-propagation algorithm.

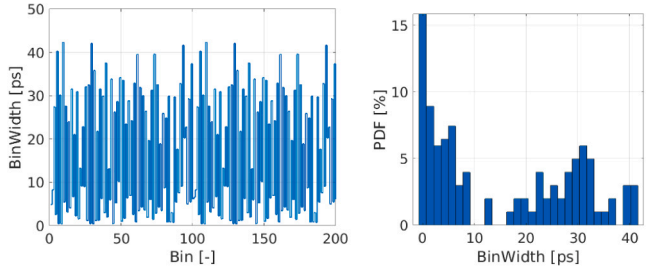


Fig. 12. *BinWidth* (left) and the corresponding distribution (right) acquired respectively in Xilinx 28-nm Artix-7 (XC7A100TFG256-2).

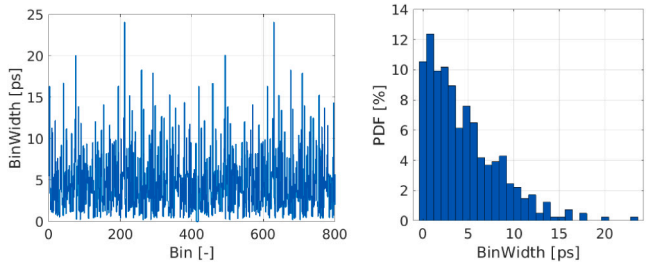


Fig. 13. *BinWidth* (left) and the corresponding distribution (right) acquired respectively in Xilinx 20-nm XCKU040-FFVA1156-2-E Kintex UltraScale.

3.2. Proposed NN

It was chosen to opt for an NN because the goal is to identify the optimal conversion curve between (pseudo) thermometric code and picoseconds; consequently, in order to be conservative, the most comprehensive calculation system was chosen, namely an NN-based approach. This choice was also dictated by the fact that the *BinValue* curves produced by classical TDL-TDCs present in the scientific literature and implemented in FPGA are highly non-linear [69]. This is induced by a strong dispersion of propagation times of the various taps, resulting in very dispersed *BinWidth* values. In reference to the size of the TDL, for technological reasons related to the size of FPGAs and the maximum clock they can support, referring to Table 1, TDLs implemented on 28-nm 7-Series FPGAs are characterized by approximately 180–220 taps [37,54], while those on 20-nm UltraScale FPGAs by approximately 800–1000 [59]. In Figs. 12 and 13, the *BinWidth* (left) and the corresponding distribution (right) acquired respectively in Xilinx 28-nm Artix-7 (XC7A100TFG256-2) and Xilinx 20-nm XCKU040-FFVA1156-2-E Kintex UltraScale are depicted.

Regarding the decoders, according to the scientific literature, whether they are Log2 [62] or sum1s types [31], they are always pipeline structures that proceed dichotomously, with an input stage having a number of single-bit inputs equal to the size of the TDL. This means 256 single-bit inputs for Xilinx 28-nm 7-Series FPGAs and 1024 for 20-nm UltraScale.

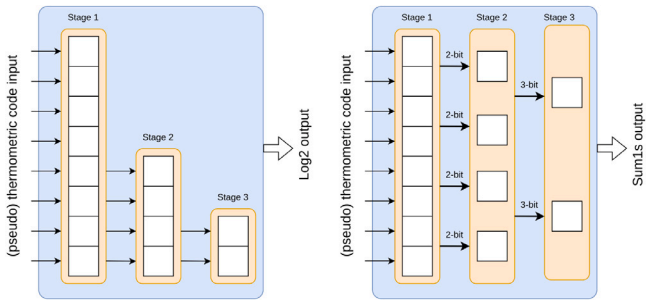


Fig. 14. Log2 (left) and sum1s (right) decoder's topologies for a 8-bit (pseudo) thermometric code.

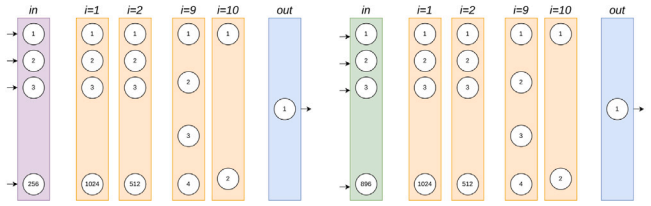


Fig. 15. Structure of the NN used for Xilinx 28-nm Artix-7 (XC7A100TFG256-2) (left) and Xilinx 20-nm XCKU040-FFVA1156-2-E Kintex UltraScale (right); note that the 10 hidden layers and the output layer are the same, while only the input layer changes, characterized by 256 and 896 nodes respectively.

The internal stages, in the case of Log2 decoders, are generally characterized by M inputs and $M/2$ outputs that propagate to the next stage the portion of the TDL where the 1-0 transition occurred. On the other hand, sum1s decoders are usually organized as tree adders, where each intermediate stage has M inputs of m bits and $M/2$ outputs of $m+1$ bits. A sketch is reported in Fig. 14. Each white block of Fig. 14 represents the basic engine element that the hardware uses to compute the Log2 or the sum1's from the M inputs and $M/2$ outputs. The sole function of this element is to propagate the binary input information to output in order to extract the *BinAddress* from the (pseudo) thermometric input code; it will then be the calibrator's task to convert the *BinAddress* (binary-type information) into time.

The basic structure of the proposed NN, identical in the hidden and output layers for both the 28-nm and 20-nm cases, closely follows the topology described above. Therefore, it consists of 10 fully connected hidden layers (since $1024 = 2^{10}$ and 10 are the pipeline stages request for decoding a 1024-long TDL) where the ith layer, with $i \in [1;10]$, is composed by 2^{10-i+1} nodes (e.g., 1024 for the first, 512 for the second, and 2 for the tenth). The input layer is composed by one node for each bit of the (pseudo) thermometric code; i.e., 256 nodes for the 28-nm technology node (Section 4) and 896 nodes for the 20-nm technology node (Section 5); this solution is chosen to adapt the (pseudo) thermometric code with different length at the same internal structure. The fully connected layers are used to emulate the reordering functionality reported in [30,66,67]. The output layer consists of only one node and is interchanged with the 2 nodes of the 10th hidden layer. The graphical representation of the proposed NNs are reported in Fig. 15.

Considering the absence of this topic in scientific literature and the multitude of degrees of freedom in NN design (e.g., number of layers, number of nodes per layer, node connectivity, weights, and activation function), it was chosen to mimic the hardware structure of the decoding mechanism in the topology of the NN illustrated above. This is clearly visible when comparing Figs. 14 and 15, where it can be seen how the engine for computing Log2 and sum1s has been replaced by a node with the corresponding activation function and weights. A leaky ReLU activation function was chosen because the task of the hardware engine is to propagate or not the binary information to the

output, similar to ReLU. The choice of leaky ReLU was made to avoid potential convergence issues. In hardware, each engine manages binary input and output information, deciding whether to propagate it forward or not. Meanwhile, the calibration task involves converting this binary information into time. In the proposed NN, since the input is a (pseudo) thermometric code (i.e., 0 or 1), it is the weights that contribute to transforming this information into time, thus serving as a distributed calibrator. For this reason, overly abrupt activation functions were not considered. In this context, as an activation function, a compromise between simplicity and performance was chosen [74], relying on the widely used leaky ReLU activation function (Deep) with an alpha value of 0.01. Moreover the weight are initialized using He algorithm with a uniform distribution.

The choice fell on Leaky ReLU rather than simple ReLU to address the Dying ReLU problem [76]. Since the neural network (NN) will act as a decoder and a distributed bin-by-bin calibrator, the nodes of the NN will need to replace the logical/mathematical operations performed by logic gates in the FPGA-implemented decoder logic. ReLU, as highlighted in [77], using the example of an XOR (i.e., a basic element of addition and thus of the sum1's decoder), suffers from the Dying ReLU problem in this scenario, which is resolved with Leaky ReLU.

The purpose of the NN is to perform the thermometric-to-time conversion recognizing stochastic BEs from deterministic ones and compensate for them improving the precision of the TDC. In this way, it will be possible to compare the precision achieved through NN with that of the main conversion algorithms known in the literature and presented in Section 2.2.2.

3.3. Proposed NN supervised learning training algorithm

The principal metric to characterize TDCs is the precision (i.e., σ_{TDC}), which measures the consistency and repeatability of a measurement [78]. One way to measure the precision is the START vs. STOP Precision (SSP),[4] which consists of the combination of standard deviations obtained averaging measures of a fixed time interval T_{meas} characterized by an intrinsic jitter $\sigma_{T_{meas}}$ between the START and STOP signals

$$SSP = \sqrt{\sigma^2_{T_{meas}} + \sigma^2_{TDC}} \tag{5}$$

In this sense, we can express the precision (σ_{TDC}) as function of different contribution obtaining,

$$SSP = \sqrt{\sigma^2_{T_{meas}} + \sigma^2_{NI} + \sigma^2_{Q,1} + \sigma^2_{Q,2} + \sigma^2_{ELN}} \tag{6}$$

where σ_{NI} is the uncertainty associated to the coarse contribution in Nutt-Interpolation due to the jitter of the clock; σ_Q is the contribution due to quantization errors [79] of the fine parts, i.e., due to the TDL-TDC; and σ_{ELN} is the total electronic jitter related to electronic components on the START and STOP channels (that can be consider negligible if high-speed electronic is used) [80]. Actually, the jitter contributions composing σ_{ELN} may be due to random jitter phenomena of the circuitry interposed between the START/STOP signals and the TDC, and to threshold jitter phenomena, defined as the ratio between the electronic voltage noise affecting the circuit (v_n) and the slope of the signals (Sl); that is, i,e. $\sigma_{ELN} = v_n/Sl$. Among these contributions, only $\sigma_{Q,1}$ and $\sigma_{Q,2}$ can be modified at the TDL-TDC's design stage, by selecting the best combination of decoder and calibration procedures, if the classical approach is adopted, or, as suggest in this paper, using a ML-based approach. The Eq. (5) does not account for the impact of TDL non-linearities on SSP, as it is assumed to work with calibrated systems. Consequently, such non-idealities are included for in $\sigma_{Q,1}$ and $\sigma_{Q,2}$ through the LSB_{EQ} presented in (4).

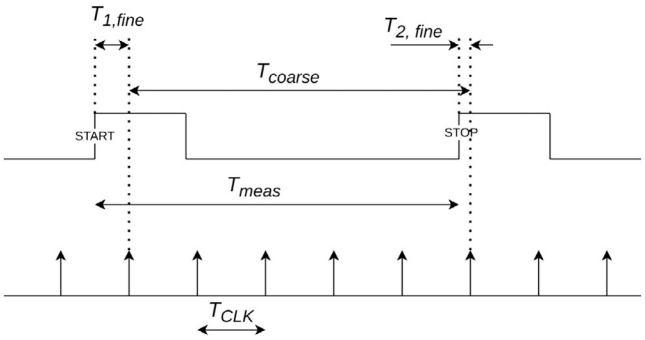


Fig. 16. Waveform of the Nutt-Interpolation considering a single TDL-TDC for the fine measurements (i.e., $T_{1,fine}$ and $T_{2,fine}$) plus a coarse counter for the coarse part (i.e., T_{coarse}).

Aim of the proposed algorithm is approximating a function $f(X,\theta)$ that maps (pseudo) thermometric codes to time values ($BinValue$), where X refers to the (pseudo) thermometric code, and θ to the parameters of the NN. A supervised learning method can build $f(X,\theta)$, but an array of inputs with corresponding desired outputs (labels) is also needed. This list of inputs and desired outputs is called training list. The selected training method is the bin inference through SSP minimization by repeating L times the measurement of the same time interval T_{meas}.

It was decided to focus on a specific scenario to determine $f(X,\theta)$ by utilizing data from the period measurement of a single channel of the Nutt-Interpolator, aiming to minimize computational overhead (i.e., only one NN is required because only one thermometric-to-time converting function is requested). In this context, as represented in Fig. 16, T_{meas} denotes the period of a signal measured by a single TDL-TDC (as fine) plus a coarse counter; in this context, the START or STOP signals that define the time interval under measurement are two consecutive rising edges, while $\sigma_{T_{meas}}$ represents cycle-to-cycle jitter. With reference to (2) and Fig. 16, $T_{2,fine}$ and $T_{1,fine}$ are the fine part of the measurement produced by the same TDL-TDC (i.e., $T_{2,fine}$ and $T_{1,fine}$ are derived from the same thermometric-to-time function, with $T_{2,fine}$ taking as input the thermometric code of the STOP edge, and $T_{1,fine}$ taking as input that of the START edge.) with a contribution σ_Q (i.e., $\sigma_{Q,1} = \sigma_{Q,2}$), while T_{coarse} represents the distance in clock cycles with precision σ_{NI}.

Under this condition (6) can be written as follow,

$$SSP = \sqrt{\sigma^2_{T_{meas}} + \sigma^2_{NI} + 2\sigma^2_{Q} + \sigma^2_{ELN}} \tag{7}$$

From (7) knowing that $\sigma_Q = LSB_{EQ}/\sqrt{12}$ we obtain,

$$SSP = \sqrt{\sigma^2_{T_{meas}} + \sigma^2_{NI} + \frac{2}{12}LSB^2_{EQ} + \sigma^2_{ELN}} \tag{8}$$

Reverting (8) we have the following inequality that become an identity only if jitters are negligible.

$$LSB_{EQ} \leq \sqrt{\frac{12}{2}} \cdot SSP \simeq 2.45 \cdot SSP \tag{9}$$

The SSP of the collected data set can be expressed as multivariate function expressed in matrix form as

$$D_{L,1} = S_{L,Therm} \cdot F_{Therm,1} + C_{L,1} \tag{10}$$

where $D_{L,1}$ is a $(L,1)$-vector containing the data-set of L values T_{meas}; $C_{L,1}$ is a $(L,1)$-vector of the coarse values T_{coarse} of each one of the L samples; $F_{Therm,1}$ a $(Therm,1)$-vector that represents the thermometric-to-time function that assigns to each (pseudo) thermometric code in the data-set (i.e., $Therm$) a time value (i.e., $BinValue[i]$ with $i \in [0; Therm-$

[4] The single-shot precision is the precision contribution offered by the single registration of one of the two START and STOP events that define T_{meas}. In the case where the two measurement channels for START and STOP events offer the same precision, it corresponds to $SSP/\sqrt{2}$.

1]). In this sense, $F_{Therm,1}$ represents decoding plus calibration product. Finally, the $S_{L,Therm}$ is a $(L, Therm)$-matrix filled with $+1$, -1, and 0, where each row is one of the L samples and each column corresponds to the contribution of each (pseudo) thermometric code in each sample referring to $T_{1,fine}$ and $T_{2,fine}$.

For example, in case of 4-tap TDL (without BE) with only $Therm = 5$ possible different thermometric codes ("0000", "1000", "1100", "1110", and "1111"), (10) becomes

$$\begin{bmatrix} T_{meas}[0] \\ T_{meas}[1] \\ \cdots \\ T_{meas}[L-1] \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 & -1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & -1 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} BinValue[0] \\ BinValue[1] \\ BinValue[2] \\ BinValue[3] \\ BinValue[4] \end{bmatrix} + \begin{bmatrix} T_{coarse}[0] \\ T_{coarse}[1] \\ \cdots \\ T_{coarse}[L-1] \end{bmatrix}$$

This means that, if we take the measurement with index 0, $T_{meas}[0]$, as an example, it will have produced a certain number of counts by the coarse counter equal to $T_{coarse}[0]$, and a pair of thermometric codes (e.g., "0000" and "1110" on the START and STOP signals respectively), corresponding to $T_{fine,1} = BinValue[0]$ and $T_{fine,2} = BinValue[3]$, resulting in a total fine contribution of $T_{fine,1} - T_{fine,2}$.

The mean value $\overline{T}_{meas}$ of the T_{meas} time interval measurements, can be calculated by multiplying the vector $D_{L,1}$ by a $(1, L)$-vector of ones (i.e., $1s_{1,L} = [1, 1, \ldots, 1]$) and normalizing by L,

$$\overline{T}_{meas} = (1s_{1,L} \cdot D_{L,1}) \cdot \frac{1}{L} \tag{11}$$

The $MeanDiff_{L,1} = T_{meas} - \overline{T}_{meas}$ results an $(L, 1)$-vector of all the samples and can be written as $MeanDiff_{L,1} = D_{L,1} - 1s_{1,L}^{T} \cdot \overline{T}_{meas}$ where ^{T} represent the operation of transposed.

Finally, the SSP can be expressed as

$$SSP = \sqrt{\frac{MeanDiff_{L,1}^{T} \cdot MeanDiff_{L,1}}{L-1}} \tag{12}$$

The SSP of a set of samples is a function that depends only on the thermometric-to-time conversion, on vector $F_{Therm,1}$. In fact, the array $S_{L,Therm}$ (fine data) and the vector $C_{L,1}$(coarse data) are fixed in the training set. In the SSP, each variable is the $BinValue$ assigned to each (pseudo) thermometric code represented by $F_{Therm,1}$. Therefore, the SSP is a multivariate function of $Therm$ number of variables. The SSP has different contributions, Eq. (7), if the thermometric-to-time values of (10) (i.e., $F_{Therm,1}$) are changed, the only contribution that affects the SSP is the quantization error (σ_Q^2) so the LSB_{EQ}. The minimum SSP (SPPmin) should correspond to the best thermometric-to-time conversion, being possible that the optimized $F_{Therm,1}$ is different from the $BinValue$ used in the initialization phase. In addition, to avoid generating of "non-acceptable solutions from a physical point of view" for the $F_{Therm,1}$ (e.g.; a constant vector $F_{Therm,1} = K\ \forall Therm$), an additional constraint has been added; namely, the joint minimization between SSP and the LSB_{EQ} obtained from the vector $F_{Therm,1}$. Indeed, we need to prevent the case in which the algorithm reduces the SSP to zero, caused by the fact that all measurements $D_{L,1}$ take the same value; i.e., $D_{L,1} = C_{L,1} + K$. The resulting vector $F_{Therm,1}$ from the NN algorithm, from a physical perspective, corresponds to the $BinValue$ of a classic TDL-TDC composed of a decoder and a bin-by-bin calibrator. Ergo, by reversing (3), considering that $F_{Therm,1}$ is $BinValue[i]$ with $i \in [0; Therm - 1]$, it is possible to deduce the temporal duration and the order of (pseudo) thermometric codes (i.e., $BinWidth[i]$ with $i \in [0; Them - 1]$), and thus the CT. Now, using Eq. (4), it is possible to calculate the LSB_{EQ}.

Indeed, in the extreme case where $F_{Therm,1}$ results a constant vector, the curve $\Delta F_{Therm,1}$ presents a peak for $Therm = 0$, maximizing LSB_{EQ}.

In summary, to train the NN, these steps should be followed.

(1) Measure a fixed time interval T_{meas} with the same TDL L times. The acquired (pseudo) thermometric codes compose the training set.
(2) Use an optimization algorithm to minimize the output of the loss function (Section 3.1 and Fig. 11) calculated as the Mean Squared Error (MSE) between the desired output T_{meas} and the output of the NN. There are many ways of multivariate function minimization [81–84]. Some use the gradients of the objective function, others use bounds for the variables, others perform local instead of global optimization, and so on. We have opted for the Adaptive Moment Estimation (ADAM) choosing beta1 = 0.9, beta 2 = 0.999, mini-batch = 64, and epoch = 20. MSE and ADAM have been chosen because they are the more common solutions in NN.
(3) The vector $F_{Therm,1}$ obtained by minimizing the training set's SSP contains the expected time value assigned to each (pseudo) thermometric code. Now, add to a training list the (pseudo) thermometric codes as inputs, and their bin values in $F_{Therm,1}$ as labels.
(4) Repeat steps 1 through 3 as many times as the amount of samples in the training list is sufficient high to recognize patterns caused by deterministic BEs and to distinguish between deterministic and stochastic BEs. We considered the training process to be concluded when the SSP values were comparable to the state-of-the-art, i.e., $10 \div 20$ ps r.m.s. for the 28-nm solution [30] and $5 \div 10$ ps r.m.s. for the 20-nm one [66]. Since LSB_{EQ} is physically linked to SSP via (9), we considered the learning process concluded when the SSP entered the above-mentioned ranges with LSB_{EQ} at least 2.7 times the SSP (i.e., 2.45 + 10% tolerance).
(5) Perform supervised learning. The NN trained with the training list is subsequently employed to predict time values from (pseudo) thermometric codes instead of the traditional decoding plus calibration procedure. Indeed, if the training, as described in steps 1 and 4, is executed successfully, the NN will be able to recognize repetitive patterns caused by deterministic BEs, compensating for them and minimizing the SSP as intended, without being negatively influenced by stochastic BEs in terms of SSP.
(6) Compare the result obtained in step 5 with the solutions from state-of-the-art classic decoders plus bin-by-bin calibration available in scientific literature.

For the learning phase, given the novelty of the subject with limited literature, and considering the various degrees of freedom (e.g., learning type and algorithm, loss function, choice of optimizer, and parameter selection), we opted for supervised learning. The ADAM optimizer was chosen because it is a common and effective optimizer that is well-suited to noisy gradients (i.e., primarily due, in our case, to the presence of electronic noise σ_{ELN} in the SSP) and complex models (i.e., BE distribution) [85]. Its effectiveness in general-purpose applications is confirmed by several publications [86,87]. Various guides and tools on the selection of ADAM hyperparameters are available [88–90], and both agree on the default values for the hyperparameters alpha (i.e., 0.001), beta1 (i.e., 0.9), beta2 (i.e., 0.999), and epsilon (i.e., 10^{-8}), which is why these values have been maintained. Instead, we worked with a mini-batch size of 64 to achieve a good compromise between gradient accuracy and computational efficiency, without focusing on convergence time in this initial investigative paper [91].

The decision for supervised learning was driven by its simplicity compared to unsupervised and reinforcement learning, but also by the

fact that, with the measurement interval T_{meas} known through (10), it was possible to assign an expected value in time to each input (pseudo) thermometric code. The same learning model was applied to both technology nodes.

4. Test and comparison in 28-nm Artix-7 FPGA

For experimental validation, the suggested method has been implemented in a Xilinx 28-nm Artix-7 (XC7A100TFG256-2) FPGA device that is hosted in an instrument (FELIX) for timing measurements [92] provided by TEDIEL S.r.l. [93] that guarantee a voltage noise v_n lower than 18 mV r.m.s. Moreover, each channel of the FELIX system has a threshold comparator characterized by a random jitter of 2 ps r.m.s. For the purpose of the paper, the FELIX system was programmed with firmware different from those stock ones advertised in the commercial product brochures. The reference TDL-TDC is a 256-taps TDL based on CARRY4 (i.e., $N = 256$) and the Nutt-Interpolation is performed with clock of a period of 2.4 ns and a jitter lower than 90 fs r.m.s., this make the σ_{NI} negligible [62]. The sum1s followed by a bin-by-bin calibrator is clearly the best decoding method, according to an analysis of the scientific literature on the 7-Series, and is thus adopted as the benchmark for the traditional approach.

In these context, two firmware were produced; one, used as a reference, with a classic TDL-TDC followed by a sum1s decoder and bin-by-bin calibrator instantiated in FPGA; the other, used as a test, in which (pseudo) thermometric codes generated by the TDL-TDC, along with the coarse measurement, were sent and stored on a PC via USB 2.0. Subsequently, offline, the data acquired by the test firmware were processed by the NN algorithm described in Section 3.2; then, once the conversion function from (pseudo) thermometric code to picoseconds was obtained, relying on Eq. (10), the timestamps were extracted for comparison with the reference firmware.

The measurement signal, a 0 V–2 V square wave, was generated using an Arbitrary Waveform Generator (AWG) series 5000 provided by ACTIVE Technologies characterize by a slope Sl of 12.5 mV/ps (i.e., an amplitude of 2 V with a rise time of 160 ps). In these conditions it is possible to estimate the σ_{ELN} equal to 3.49 ps r.m.s. considering a contribution of 2.83 ps r.m.s. for the random jitter (i.e., 2 ps r.m.s. for both START and STOP) and a contribution of 2.04 ps r.m.s. for the threshold (i.e., 18 mV/12.5 mV/ps for both START and STOP). Referring to (7), AWG-5000 was used to generate the periodic signal with period T_{meas} characterized by a jitter $\sigma_{T_{meas}} < 2$ ps r.m.s., and supplied to the various firmware; so following (5), the precision offered by the two TDL-TDCs (references vs test) from the SSP is extracted. A scheme of the measurement setup is reported in Fig. 17. Given that the periodic signal generated by AWG-5000 is entirely uncorrelated with the clock of the FELIX used in the Nutt-Interpolation, it follows that the distribution of fine measurements, hence the thermometric codes acquired by the TDL, tends towards a uniform distribution [61]; this allowed us to have an appropriate dataset to use in the neural network (NN).

4.1. Experimental results

Firstly, the test firmware is used to estimate the number of samples, denoted as L, for setting up the trending list of the NN. For this purpose, a continuous number of T_{meas} measurements were performed until saturation of the (pseudo) thermometric codes uniqueness, number marked as $Therm$, was observed. Green curve in Fig. 18 (with ordinate on the left axis) shows that $Therm$ saturates at 170[5] with a number of about

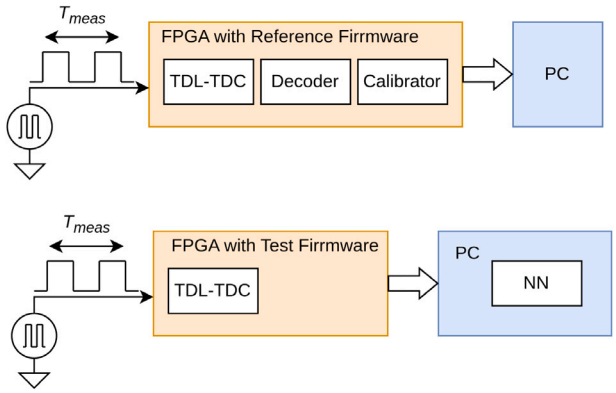


Fig. 17. Graphic representation of the measurement setup, (top) reference firmware connected to the PC for data readout, and (bottom) test firmware with the proposed NN algorithm onboard on the PC.

$L = 3 \cdot 10^5$ performed measurements; this means that the statistics of collecting (pseudo) thermometric codes can be considered exhaustive. Subsequently, the reference firmware (i.e., real-time fully FPGA-based system with sum1s decoder followed by bin-by-bin calibrator) was running in the device and we noticed that the univocal $BinAddress$ values at calibrator input saturated at 154, due to BEs and merging effect of the sum1s decoder (red curve in Fig. 18 with ordinate on the left axis). The gap of $170 - 154 = 16$ between univocal (pseudo) thermometric codes and decoded ones means that, due to BEs, at least 16 codes over 170 (i.e., 9.4%) are confused with others during the sum1s decoding process, resulting in errors of the estimation of the $BinWidth$ thus an increase in quantization error with respect to the optimal case.

Moreover, the standard deviation of the measurement performed in real-time by the reference firmware (blue curve in Fig. 18 with ordinate on the right axis) stabilized at the constant value 13.3 ps r.m.s. corresponding to $Therm$ and $BinAddress$ saturation. In conditions of saturation of both these values, we have thus confirmed $L = 3 \cdot 10^5$ to be the minimum number of measurements necessary for training the NN. In this way, the NN was trained using 10 measured sets of about $3 \cdot 10^5$ samples.

The output of NN is the thermometric-to-time conversion function from the 170 (pseudo) thermometric codes, acquired with the test firmware, to picoseconds. This conversion function was used to plot Figs. 19 and 20.

To highlight the extent of "bin merging", the BEs frequency of the (pseudo) thermometric code acquired with the test firmware was investigated. Fig. 19 shows that MBD was at most 3 and only a minimal percentage of the codes is affected, less than 10%, to be compared with the 9.4% of the BE merging effect observed in Fig. 18.

The scatter plot in Fig. 20 was created by comparing the thermometric-to-time conversion map of the 170 (pseudo) thermometric codes, acquired with the test firmware, with respect to the $BinWidth$ estimated by the referenced firmware and used to generate the $BinValue$ curve. In this sense, such a scatter plot can be considered as a comparison metric between the test firmware and the reference one, namely between the proposed ML approach (NN described in Section 3) and the classical one (sum1s with bin-by-bin real-time calibrator in FPGA present in the state-of-the-art).

The scatter plot represents a mapping between the 154 $BinAddress$ outputs from the sum1s decoder of the reference firmware and the 170 (pseudo) thermometric codes acquired by the test firmware and fed into the NN. In this context, each $BinAddress$ that has undergone a "bin merging" effect is decomposed into its corresponding 2, 3, or 4 components of (pseudo) thermometric code. Additionally, for each $BinAddress$ that has undergone "bin merging" as depicted in the figure,

[5] The total number of (pseudo) thermometric codes does not saturate at the maximum bin value of the bins (i.e., 256) because, in order to ensure proper synchronization between the TDL-TDC and the coarse counter, the implemented TDL introduces a slightly longer delay than the 2.4 ns clock period used to clock the coarse counter.

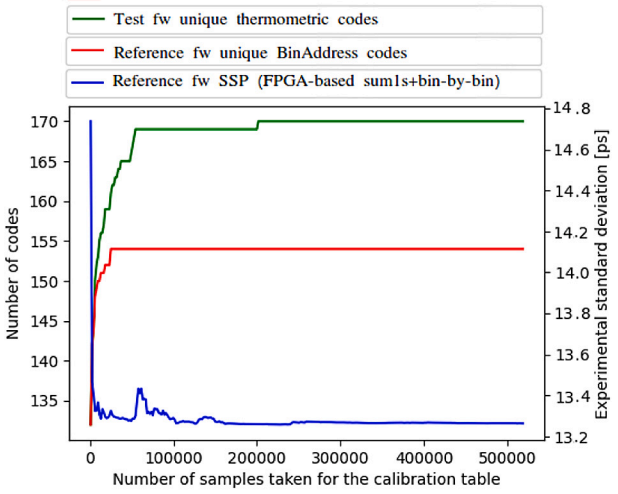


Fig. 18. Number of univocal (pseudo) thermometric codes (green) acquired with the test firmware, *binAddress* obtained by sum1s decoding (red), and SSP after bin-by-bin calibration (blue) on the reference firmware. The x-axis represents the L number of performed measurements. The left y-axis represents the number of codes reached by the test/reference firmware, while the right y-axis indicates the SSP values in ps r.m.s. achieved by the reference firmware.

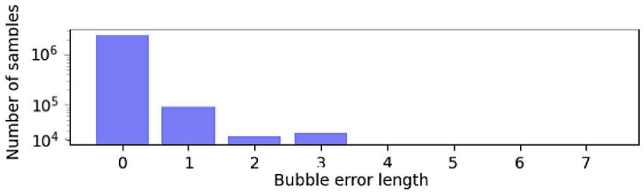


Fig. 19. Histograms of bubble error length values measured in the generated (pseudo) thermometric codes on the test firmware in 28-nm 7-Series Xilinx Artix-7 (CARRY4 primitive).

the quantization contribution offered by the two slowest (pseudo) thermometric codes (i.e., bigger in therm of propagation delay) subject to "bin merging" has been represented on the x and y axes of the plot.

We positioned the slowest contribution (BiggestBin in the figure), expressed in picoseconds, on the x-axis, and the secondary contribution (secondBiggestBin in the figure) on the y-axis. As evident from the scales on both axes, each *BinAddress* has a second contribution that is 10 times faster (i.e., smaller in therm of propagation delay) than the first slowest one. Therefore, in terms of quantization error, the (pseudo) thermometric code responsible for the second contribution is negligible.

The SSP achieved by the test firmware was then compared to that of the reference firmware (i.e., 13.3 ps r.m.s. as shown in blue in Fig. 18) by comparing the histograms and their respective standard deviations obtained from a set of $L = 3 \times 10^5$ samples, as depicted in Fig. 21. With the NN, we achieved a very similar and better SSP value (i.e., 13.1 ps r.m.s.). The K samples used to evaluate the SSP in the test and reference firmwar are obviously different from the $10 \times L$ used for training the NN to demonstrate that no underfitting and overfitting issues are present.

From a purely mathematical standpoint, since the mechanism of coarse attribution is common to both, the only difference in the measured value between the test firmware and the reference firmware, as referred to in (10), lies in the calculation of the fine component (i.e., $S_{L,Therm} \cdot F_{Therm,1}$). Indeed, concerning the test firmware, $F_{Therm,1}$ corresponds to the thermometric-to-time conversion function resulting from the NN, while for the reference firmware, $F_{Therm,1}$ corresponds to the $BinValue$ values extracted from the sum1s decoder and bin-by-bin calibration mechanism residing in the FPGA.

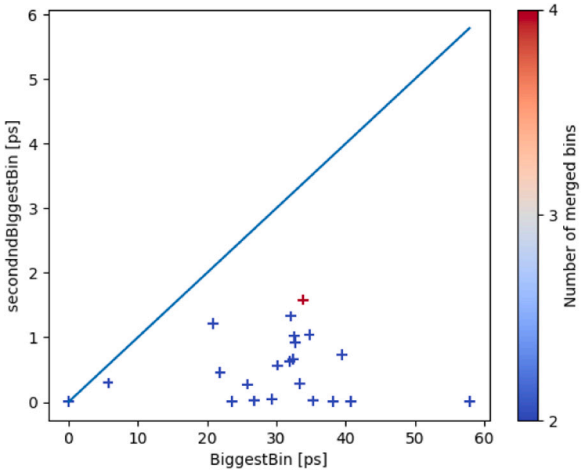


Fig. 20. Scatter plot of the merged bin's width composition observed on the 28-nm 7-Series Xilinx Artix-7 (CARRY4 primitive).

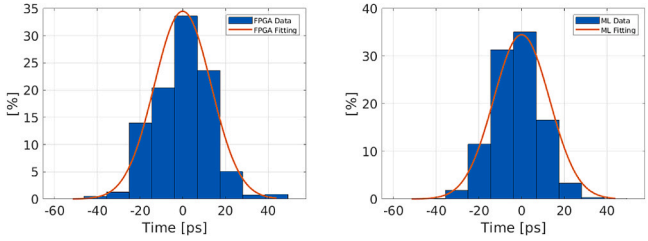


Fig. 21. SSPs obtained with the reference firmware 13.3 ps r.m.s. (left) and the test firmware 13.1 ps r.m.s. (right); in blue, the histogram of the actual data centered at zero, and in red, the corresponding fitting.

In order to prevent the presence of underfitting and overfitting issues dependent on T_{meas}, SSP measurements were repeated on both the reference and test firmware (without retraining the NN), varying the frequency of T_{meas} for 6 different values (i.e., from 775 kHz to 800 kHz with a step of 5 kHz), yielding identical results. This allowed us to verify that the above-mentioned performances are independent of the fine contribution value (i.e., $T_{1,fine} - T_{2,fine}$). Indeed, from (2), considering the contribution coarse (i.e., $floor(T_{meas}/T_{CLK})$ where T_{CLK} is the 2.4 ns period that feeds the Nutt-Interpolation), resulting in 536, 534, 530, 527, 523, 520 clock pulses for the coarse part, we can extract also the fine contribution (i.e., $T_{1,fine} - T_{2,fine} = T_{meas} - T_{coarse}$) which results in 63%, 19%, 79%, 43%, 11%, 83% of T_{CLK} for the 6 different values tested above.

4.2. Discussions

From the SSP results obtained in Section 4.1 on the test firmware (i.e., 13.1 ps r.m.s) and the reference firmware (i.e., 13.3 ps r.m.s), considering the sample jitter $\sigma_{T_{meas}}$ equal to 2 ps r.m.s., it was possible to extract the precision σ_{TDC} of the two TDCs, resulting in 12.9 ps r.m.s. (i.e., $\sqrt{13.1^2 - 2^2}$ ps r.m.s.) and 13.1 ps r.m.s. (i.e., $\sqrt{13.3^2 - 2^2}$ ps r.m.s.) respectively. Moreover, considering the contributions σ_{ELN} and σ_{NI} negligible, it was possible to extract, using (7), the quantization contribution offered by the TDL (i.e., $\sigma_Q = \sqrt{(SSP^2 - \sigma^2_{T_{meas}})/2}$) and then derive the LSB_{EQ} using the definition (i.e., $LSB_{EQ} = \sqrt{12}\sigma_Q$) that result be 31.7 ps and 32.2 ps respectively. By comparing these two $\sqrt{12}\sigma_Q$ values (i.e., LSB_{EQ} derived from (7)), it is possible to notice a difference of only 0.5 ps (or 5.65 ps r.m.s.), attributable to the reduced "bin merging" phenomenon (9.4%) detected in the reference firmware. This reduced phenomenon explains the similar precision obtained by the two approaches.

Table 2
Comparison of the SSP, σ_{TDC}, σ_Q, LSB_{EQ} derived from (7) (i.e., $\sqrt{12}\sigma_Q$) and LSB_{EQ} computed using (4) obtained with the reference firmware (i.e., sum1s plus bin-by-bin calibrator) versus test one (i.e., ML-based thermometric-to-time conversion). An overestimation of the term $\sqrt{12}\sigma_Q$ compared to LSB_{EQ} is justified by neglecting σ_{ELN} (9).

Method	SSP	σ_{TDC}	σ_Q	$\sqrt{12}\sigma_Q$	LSB_{EQ} (4)
sum1s+bin-by-bin [ps r.m.s]	13.3	13.15	9.30	32.2	32.1
ML-based [ps r.m.s]	13.1	12.95	9.16	31.7	31.6

Table 3
Comparison of the σ_{TDC} obtained with the classical FPGA approach (i.e., sum1s or reordering followed by Log2 decoders plus bin-by-bin calibrator) versus ML-based thermometric-to-time conversion.

Method	σ_{TDC}	Ref.
sum1s+bin-by-bin [ps r.m.s]	13.1	This work
reorder+Log2+bin-by-bin [ps r.m.s]	12.7	[30]
ML-based [ps r.m.s]	12.9	This work

The substantial difference between the test firmware and the reference firmware is the mechanism of thermometric-to-time conversion. From Fig. 18., it is inferred that for the test firmware, 170 (pseudo) thermometric codes are identified at the TDL output, and thanks to the NN mechanism, we achieve a thermometric-to-time conversion curve that utilizes all these codes, resulting in an LSB_{EQ} (i.e., computed using (4)) of 31.6 ps. Conversely, for the reference firmware, the sum1s decoder manages to extract only 154 unique codes out of the 170 available, and after the calibration process, we obtain a bin width curve characterized by an LSB_{EQ} (i.e., computed using (4)) of 32.1 ps. Also in this case, a small difference of only 0.5 ps (or 5.64 ps r.m.s.) between the two LSB_{EQ} is present. The 16 missing codes in the reference firmware underwent a "bin merging" phenomenon, where the decoder mistakenly interpreted two or more different pseudo thermometric codes with the same number of 1s due to the BE (Fig. 20.). The phenomenon is mitigated in terms of SSP since the BE is small (Fig. 19) to plot Figs.

Results are summarized in Table 2.

In addition, the results obtained in this study are directly comparable to the precision of 12.7 ps r.m.s. reported in the scientific literature, which was achieved using reordering, Log2 decoding, and bin-by-bin calibration in Xilinx 28-nm FPGA [30]. This indicates that employing a sum1s decoder followed by a bin-by-bin calibrator for the conversion of (pseudo) thermometric code to time measurements is the optimal approach for implementing the TDL based on the primitive CARRY4. Furthermore, the significant disparity in the sizes of contributions suggests that merging occurs as a consequence of BEs caused by statistical effects, which are indeed rare events (see Table 3).

5. Test and comparison in 20-nm Kintex UltraScale FPGA

The same characterization road map has been carried out with the implementation in a Xilinx 20-nm XCKU040-FFVA1156-2-E Kintex UltraScale of a KCU105 evaluation kit for high-speed processing (i.e., v_n lower than 10 mV r.m.s.). The implemented TDL-TDC, provided by TE-DIEL S.r.l. [93], consists of a TDL based on CARRY8 fabric primitives, with 1024 taps ($N = 1024$) sampled with the same clock of 2.4 ns and jitter lower than 90 fs (i.e., negligible σ_{NI}). The measurement signal, a 0 V–2 V square wave, was generated using the AWG-5000 with a slope SI of 12.5 mV/ps and applied directly to the FPGA. Under these conditions, it is possible to estimate the σ_{ELN} of 1.13 ps r.m.s. due only to the threshold jitter (i.e., 10 mV/12.5 mV/ps for both START and STOP). The period under measure is always characterize by a cycle-cycle jitter of 2 ps r.m.s. For experimental validation, we followed the same approach outlined in Section 4. The KCU105 was programmed with two different firmwares: one served as a reference, implementing a classic TDL-TDC followed by a sum1s decoder and bin-by-bin calibrator, while the other was used as a test, where the (pseudo) thermometric codes generated by the TDL-TDC, along with the coarse measurement, were transmitted and stored on a PC via USB 3.0.

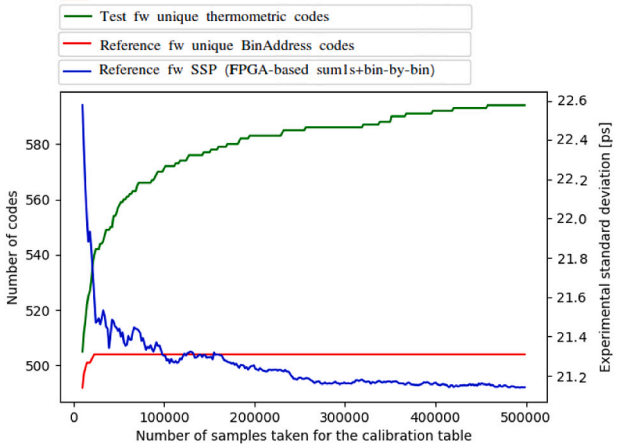


Fig. 22. Number of univocal (pseudo) thermometric codes (green) acquired with the test firmware, *BinAddress* obtained by sum1s decoder (red), and SSP after bin-by-bin calibration (blue) on the reference firmware. The x-axis represents the L number of performed measurements. The left y-axis represents the number of codes reached by the test/reference firmware, while the right y-axis indicates the SSP values in ps r.m.s. achieved by the reference firmware.

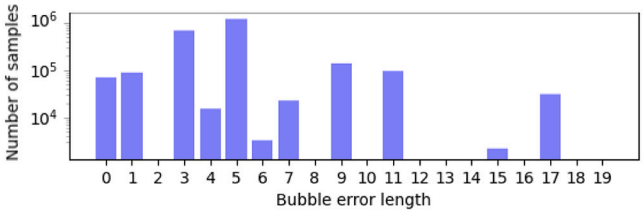


Fig. 23. Histogram of bubble error length observed on the raw thermometric codes on the test firmware in 20-nm Xilinx Kintex UltraScale (CARRY8 primitive).

5.1. Experimental results

Fig. 22 is the equivalent of Fig. 18. The (pseudo) thermometric codes of test firmware saturate at $Therm = 590$ and the *BinAddress* of reference one at 505, showing a gap of 85 bins, which means a "bin merging" due to the BEs equal to 14.4%, decidedly higher than the 9.4% obtained in the case of the use of the CARRY4. The overall saturation of $Therm$ and *BinAddress* occurs for $L = 4.5 \cdot 10^5$ measurements. The SSP of the reference saturates at 21.0 ps r.m.s.

The output of the NN is the thermometric-to-time conversion function from the 590 (pseudo) thermometric codes, acquired with the test firmware, to picoseconds. This conversion function was used to plot Figs. 23 and 24, which clearly show the severity of the BE effect compared to the 7-Series and how the sum1s decoder is not effective.

In details, Fig. 23 shows a MBD equal to 17 that is 5.67 times more with respect to the MBD of 3 in Fig. 19.

Instead, unlike what happens for the Artix-7, using the data adopted to generate Figs. 20 and 24, we have computed that almost the 30% of the *BinAddress* (i.e., 152 over the total 505) of the reference firmware is affected by "bin merging"; moreover almost the 40% of the merged bins (i.e., 62 over 152) offer a contribution of the second biggest bin that no more negligible (i.e., being bigger than 1/10 of the principal one). This is due to the fact that the sum1s algorithm is not optimal to solve deterministic BEs in this technology.

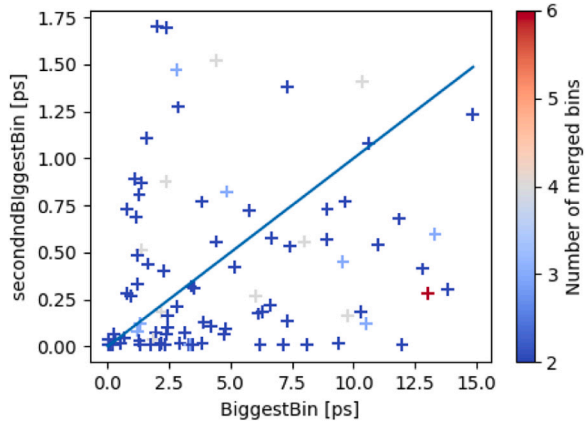


Fig. 24. Scatter plot of bin width between merged bins observed in the 20-nm Xilinx Kintex UltraScale (CARRY8).

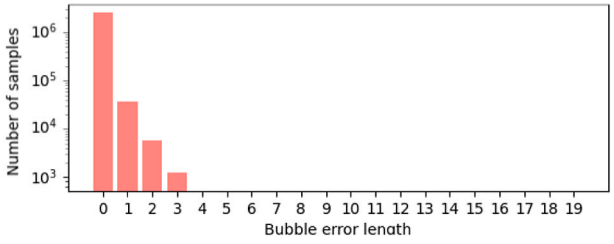


Fig. 25. Histogram of bubble error length after reordering.

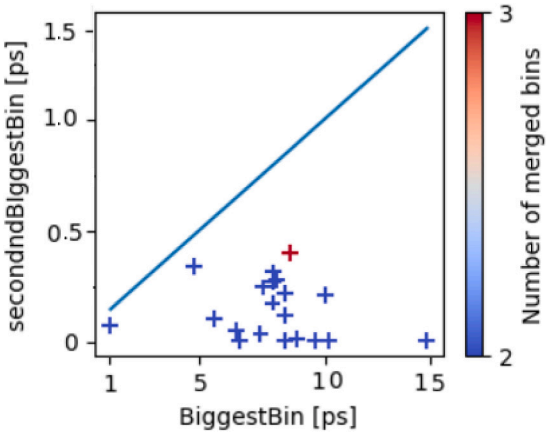


Fig. 26. Scatter plot of bin width between merged bins after reordering.

Analyzing the thermometric-to-time function obtained from NN, it is possible to observe a benefit on MBD obtained by reordering the taps as proposed in [66], whose results are illustrated in Figs. 25 and 26. In Fig. 25, it is possible to observe that the MBD is reduced up to 3 and only the 10% of pseudo thermometric codes are affected by BE. Similarly, from the scatter plot of Fig. 26, it is possible to see that the BE effect becomes negligible thanks to reordering, as in Artix-7. In this term, we have demonstrated that the better choice to real-time processing in FPGA invoke the use not only of the sum1s decoder and the bin-by-bin calibrator but also to a reordering mechanism followed by Log2 decoder to compensate the BE due to deterministic issues (i.e, routing and clock skew).

The SSP achieved by the test firmware was then compared to that of the reference firmware (i.e., 21.0 ps r.m.s. as shown in blue Fig. 22) by comparing the histograms and their respective standard deviations obtained from a set of $L = 4.5 \times 10^5$ samples, as depicted in Fig. 27. With the NN, we achieved a better SSP of 5.24 ps r.m.s. which is very similar to the SSP of 5.28 ps r.m.s. achieved with reordering mechanism followed by Log2 decoder. The L samples used to evaluate the SSP in the test firmware are obviously different from the $10 \times L$ used for training the NN to demonstrate that no underfitting and overfitting issues are present.

Similarly, here, since the mechanism of coarse attribution is common to both, the only difference in the measured value between firmwares, as referred to in (10), lies in the calculation of the fine component (i.e., $S_{L,Therm} \cdot F_{Therm,1}$). Indeed, concerning the test firmware, $F_{Therm,1}$ corresponds to the thermometric-to-time conversion function resulting from the NN, while for the sum1s and reordering followed by Log2 decoders plus bin-by-bin calibrator, $F_{Therm,1}$ corresponds to the $BinValue$ values extracted from the decoders and calibration mechanisms residing in the FPGA.

The SSP measurements were repeated for different values of T_{meas} (i.e., from 775 kHz up to 800 kHz with a step of 5 kHz) in order to expose fine contributions of varying values (i.e., 63%, 19%, 79%, 43%, 11%, 83% of T_{CLK}) obtaining similar results.

5.2. Discussions

From the SSP results obtained in Section 5.1 on the test firmware (i.e., 5.24 ps r.m.s), the reference firmware (i.e., 21.0 ps r.m.s) and the reference firmware where the sum1s decoder is replaced with reordering followed by Log2 (i.e., 5.28 ps r.m.s), considering the sample jitter $\sigma_{T_{meas}}$ equal to 2 ps r.m.s., it was possible to extract the precision σ_{TDC} of the three TDCs, resulting in 20.9 ps r.m.s. (i.e., $\sqrt{21.0^2 - 2^2}$ ps r.m.s.), 4.89 ps r.m.s. (i.e., $\sqrt{5.28^2 - 2^2}$ ps r.m.s.), and 4.85 ps r.m.s. (i.e., $\sqrt{5.24^2 - 2^2}$ ps r.m.s.) respectively. Moreover, considering the contributions σ_{ELN} and σ_{NI} negligible, it was possible to extract, using (7), the quantization contribution offered by the TDL (i.e., $\sigma_Q = \sqrt{(SSP^2 - \sigma^2_{T_{meas}})/2}$) and then derive the LSB_{EQ} using the definition (i.e., $LSB_{EQ} = \sqrt{12}\sigma_Q$) that result be 51.3 ps, 12.0 ps, and 11.9 ps respectively. By comparing these three $\sqrt{12}\sigma_Q$ (i.e., LSB_{EQ} derived from (7)), it is possible to notice a difference of only 0.1 ps (or 1.55 ps r.m.s.) between reordering followed by Log2 and the ML approach, attributable to the reduced "bin merging" phenomenon. This reduced phenomenon explains the similar precision obtained by the two approaches. Unlike what happens between sum1s and the ML-based approach where a difference between $\sqrt{12}\sigma_Q$ of 39.4 ps (or 49.9 ps r.m.s.) is observed.

Also in the UltraScale the substantial difference between firmwares is the mechanism of thermometric-to-time conversion. From Fig. 22., it is inferred that for the test firmware, 590 (pseudo) thermometric codes are identified at the TDL output, and thanks to the NN mechanism, we achieve a thermometric-to-time conversion curve that utilizes all these codes, resulting in an LSB_{EQ} (i.e., computed using (4)) of 11.5 ps. Conversely, for the sum1s decoder manages to extract only 505 unique codes out of the 590 available, and after the calibration process, we obtain a bin width curve characterized by an LSB_{EQ} (i.e., computed using (4)) of 51.1 ps. Also in this case, a difference of 39.6 ps (or 49.8 ps r.m.s.) between the two LSB_{EQ} is present. The 85 missing codes underwent a "bin merging" phenomenon, where the sum1s decoder mistakenly interpreted two or more different pseudo thermometric codes with the same number of 1s due to the high BEs (Figs. 23 and 24) ruining the SSP. On the other hand, BEs are mitigated thanks to reordering followed by Log2 decoder (Fig. 25) reducing the "bin merging" phenomena (Fig. 25.) achieving an LSB_{EQ} (i.e., computed using (4)) of 11.7 thus a state-of-the-art SSP. Also in this case, a difference of 0.2 ps (or 2.15 ps r.m.s.) between the two LSB_{EQ} is present.

Results are summarize in Table 4.

Furthermore, we have compared the precision of 4.85 ps r.m.s achieved with reordering followed by Log2 plus bin-by-bin calibrator with

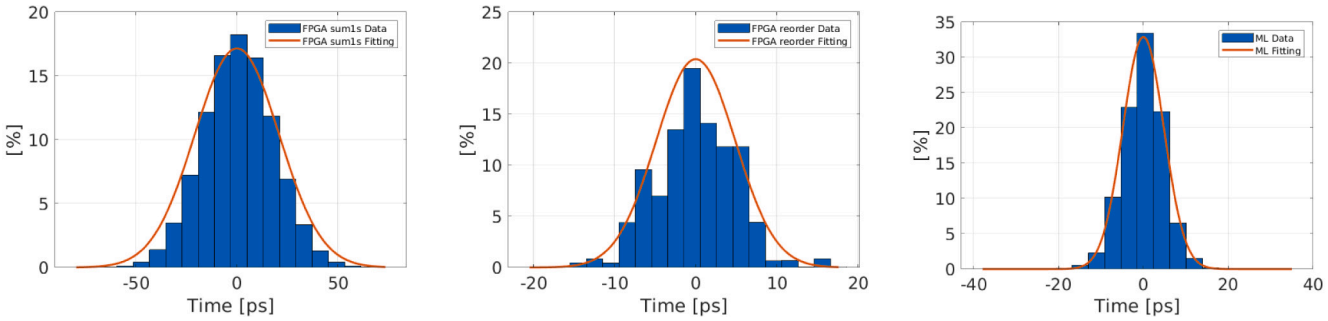


Fig. 27. SSPs obtained with the reference firmware 21 ps r.m.s. (left), reordering 5.28 ps r.m.s. (center), and the test firmware 5.24 ps r.m.s. (right); in blue, the histogram of the actual data centered at zero, and in red, the corresponding fitting.

Table 4
Comparison of the SSP, σ_{TDC}, σ_Q, LSB_{EQ} derived from (7) (i.e., $\sqrt{12}\sigma_Q$), and LSB_{EQ} obtained with the reference firmwares (i.e., sum1s or reordering followed by Log2 decoders plus bin-by-bin calibrator) versus test one (i.e., ML-based thermometric-to-time conversion). An overestimation of the term $\sqrt{12}\sigma_Q$ compared to LSB_{EQ} is justified by neglecting σ_{ELN} (9).

Method	SSP	σ_{TDC}	σ_Q	$\sqrt{12}\sigma_Q$	LSB_{EQ} (4)
sum1s+bin-by-bin [ps r.m.s]	21.0	20.9	14.8	51.3	51.1
reorder+Log2+bin-by-bin [ps r.m.s]	5.28	4.89	3.46	12.0	11.7
ML-based [ps r.m.s]	5.24	4.85	3.43	11.9	11.5

Table 5
Comparison of the σ_{TDC} obtained with the classical FPGA approach (i.e., sum1s or reordering followed by Log2 decoders plus bin-by-bin calibrator) versus ML-based thermometric-to-time conversion.

Method	σ_{TDC}	Ref.
sum1s+bin-by-bin [ps r.m.s]	20.93	This work
reorder+Log2+bin-by-bin [ps r.m.s]	4.89	This work
reorder+Log2+bin-by-bin [ps r.m.s]	4.7÷5.6	[66]
ML-based [ps r.m.s]	4.85	This work

that reported in the scientific literature [66] on the same hardware that is in the range from 4.7 ps r.m.s up to 5.6 ps r.m.s confirming the coherence with respect to the state-of-the-art. Thus, thank to the ML we have demonstrated a significantly reduction of the quantization error induced by deterministic BEs due to the reordering approach (see Table 5).

6. Future developments

Not finding anything similar in the scientific literature and considering the numerous degrees of freedom in the design of an NN and in the training process (Section 3.1), we decided to emulate the hardware topology in the software NN structure (Section 3.2) and to follow a classic supervised learning approach (Section 3.3). This decision led us to the structure described above and used. While aware that the NN structure does not necessarily have to mirror the topology of the hardware decoding and calibration mechanism, we chose to use the proposed NN because, by analyzing the thermometric-to-time conversion curve, we found a consistent explanation for the phenomenon of "bin merging" and BE present in TDL, as illustrated in Sections 4 and 5, thus achieving our goal. Further analysis and optimizations of the NN will be carried out in future developments. In addition to redesigning the NN and varying the activation function, other noteworthy future developments include the evaluation of other optimization algorithms as replacements for ADAM [94], the tuning of the dropout parameter [95], and the use of algorithms for automatic hyperparameter tuning [96].

7. Conclusions

FPGAs do not have tunable delay-lines as native resources and, consequently, FPGA-based TDL-TDC must be implemented with alternative structures, such as the CARRY primitive, devoted to carry look-ahead implementation. Because these elements cannot be finely tuned, they suffer from non-idealities and other effects, such as BEs (i.e., deterministic and statistic) and PVT. In a real-time implementation on modern FPGAs, PVT variations can be compensated for through bin-by-bin calibration of the binary code obtained by appropriate decoding of the thermometric code recorded by the TDL. Unfortunately, BEs produce pseudo thermometric codes that can cause the "bin merging" effect, where different (pseudo) thermometric codes are translated into the same binary, resulting in an increased quantization error. Moreover, due to statistical contributions to BEs, it is unfeasible to determine a priori the best choice from those available in scientific literature of decoding that minimize this effect.

The main contribution of this work lies in the strategy for decoding and calibrating the TDL using a NN. To the best of our knowledge, this approach has not been previously explored in the literature and has achieved results comparable to the state-of-the-art. Specifically, the paper demonstrates that the decoding technique, due to BEs, leads to "bin merging" phenomena identified through the NN. The ML approach was simply the tool employed.

It was decided to address the problem by developing a solution using a NN (run in Python on a PC) trained through supervised learning, minimizing the SSP (i.e., quantization error) with a fast-converging gradient method (SSPmin). Due to the absence of prior experiments, the NN was constructed based on the hardware structure of the decoders and calibrators, optimizing it with an ADAM algorithm. This approach was applied to two case studies involving 28-nm HKMG Artix-7 and 20-nm FinFET Kintex UltraScale FPGAs using a 10 fully connected hidden layers all with the leaky ReLU activation function (Deep), with 256 nodes for the Artix-7 and 896 nodes for the UltraScale. The purpose of the NN is to perform the thermometric-to-time conversion recognizing stochastic BEs from deterministic ones and compensate for them improving the precision.

The proposed methodology has been tested on the two main hardware primitives used for TDL implementation in Xilinx FPGAs, the CARRY4 and CARRY8 available in the 28-nm 7-Series and 16/20-nm UltraScale/UltraScale+ respectively; thus, the results have been compared to state-of-the-art fully FPGA-based solutions where decoders and calibrators are directly implemented in FPGA. The Artix-7 XC7A100TFG256-2 was chosen as the test FPGA for the 7-Series, and the Kintex XCKU040-FFVA1156-2-E for the UltraScale series. For TDLs based on CARRY4, a minimal number of BEs (i.e., MBD of 3) and a low "bin merging" rate (i.e., < 10%) were observed. This observation indicates that the decoding approach based on sum1s, followed by bin-by-bin calibration, achieves nearly the same precision (i.e., 13.1 ps r.m.s.) as NN approach (i.e., 12.9 ps r.m.s.), with a negligible deviation, thus validating its correctness. Moreover, the NN achieve a

precision comparable to the state-of-the-art on the same technological node [30] (i.e., 12.7 ps r.m.s.) where the deterministic BEs are corrected using reordering. The only notable difference lies in a sub-picosecond error, which, for certain applications with low area and/or high channel counts, does not justify the increased complexity associated with the reordered approach.

On the other hand, for TDLs based on CARRY8, a significant number of BEs (i.e., MBD of 17) was observed, leading to a non-negligible "bin merging" rate (i.e., 30%). This renders the decoding based on sum1s ineffective, providing a precision of 20.93 ps r.m.s. instead of the 4.85 ps r.m.s. achieved with NN. However, it has been demonstrated that reordering the bins before decoding allows reducing the MBD to 3, limiting the effects of BEs only to stochastic ones and achieving a precision of 4.89 ps r.m.s. in line with NN and the state-of-the-art on the same technological node [66] (i.e., $4.7 \div 5.6$ ps r.m.s.). In this case, the reordered approach is fundamental to guarantee high-precision.

There is nothing to prevent loading the thermometric conversion curve processed by the NN into FPGA (specifically, only its output rather than the entire NN) instead of the $BinValue$ curve. However, this step was not performed in the paper. In such a case, an area occupation comparable to the classical full-FPGA approach (i.e., decoder plus bin-by-bin calibration) would be expected.

CRediT authorship contribution statement

Fabio Garzetti: Investigation. **Nicola Lusardi:** Writing – original draft. **Enrico Ronconi:** Data curation. **Andrea Costa:** Validation. **Angelo Geraci:** Supervision.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Nicola Lusardi reports equipment, drugs, or supplies was provided by TEDIEL S.r.l. Nicola Lusardi reports a relationship with TEDIEL S.r.l. that includes: equity or stocks. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The data that has been used is confidential.

Acknowledgment

The authors would like to thanks TEDIEL S.r.l. for the FELIX instrument, the TDC IP-Core and the support.

References

[1] D. Li, et al., An 8-ch LIDAR receiver based on TDC with multi-interval detection and real-time *In Situ* calibration, IEEE Trans. Instrum. Meas. 69 (7) (2020) 5081–5090.
[2] V. Unnikrishnan, et al., Time-mode analog-to-digital conversion using standard cells, IEEE Trans. Circuits Syst. I. Regul. Pap. 61 (12) (2014) 3348–3357.
[3] P. O'Keeffe, et al., A photoelectron velocity map imaging spectrometer for experiments combining synchrotron and laser radiations, Rev. Sci. Instrum. 82 (3) (2011) 033109.
[4] L. Stebel, et al., Time-resolved soft x-ray absorption setup using multi-bunch operation modes at synchrotrons, Rev. Sci. Instrum. 82 (12) (2011) 123109.
[5] F. Garzetti, et al., Fully FPGA-based and all-reconfigurable TDC for 3D (X, Y, t) cross delay-line detectors, in: 2018 IEEE Nuclear Science Symposium and Medical Imaging Conference Proceedings, NSS/MIC, 2018, pp. 1–3.
[6] N. Lusardi, et al., Advanced system in FPGA for 3D (x, y, t) imaging with cross delay-lines, in: 2019 IEEE Nuclear Science Symposium and Medical Imaging Conference, NSS/MIC, 2019, pp. 1–4.
[7] Y. Wu, et al., A digital PLL with a multi-delay coarse-fine TDC, in: 2011 NORCHIP, 2011, pp. 1–4.
[8] Edinburgh Photonics, What is tcspc, 2012, [Online]. Available: https://www.edinst.com/wp-content/uploads/2015/08/TN2-What-is-TCSPC-Red.pdf.
[9] N. Lusardi, et al., Single photon counting through multi-channel TDC in programmable logic, in: 2016 IEEE Nuclear Science Symposium, Medical Imaging Conference and Room-Temperature Semiconductor Detector Workshop, NSS/MIC/RTSD, 2016, pp. 1–4.
[10] M. Conti, et al., The new opportunities for high time resolution clinical TOF PET, Clin. Transl. Imaging 7 (2019) 1–9.
[11] E. Venialgo, et al., Toward a full-flexible and fast-prototyping ToF-PET block detector based on TDC-on-FPGA, IEEE Trans. Radiat. Plasma Med. Sci. 3 (5) (2019) 538–548.
[12] F. Garzetti, et al., Plug-and-play TOF-pet module readout based on TDC-on-FPGA and gigabit optical fiber network, in: 2019 IEEE Nuclear Science Symposium and Medical Imaging Conference, NSS/MIC, 2019, pp. 1–4.
[13] F. Yuan, CMOS time-mode circuits and systems, in: B. Raton (Ed.), CRC Press, 2016.
[14] G. Acconcia, et al., A 1.9 ps-rms precision time-to-amplitude converter with 782 fs LSB and 0.79 rms DNL, IEEE Trans. Instrum. Meas. 72 (2023) 1–11.
[15] S. Tancock, et al., A review of new time-to-digital conversion techniques, IEEE Trans. Instrum. Meas. 68 (10) (2019) 3406–3417.
[16] R. Machado, et al., Recent developments and challenges in FPGA-based time-to-digital converters, IEEE Trans. Instrum. Meas. 68 (11) (2019) 4205–4221.
[17] F. Garzetti, et al., Assessment of the bundle SNSPD plus FPGA-based TDC for high-performance time measurements, IEEE Access 10 (2022) 127894–127910.
[18] N. Lusardi, et al., High-resolution imager based on time-to-space conversion, IEEE Trans. Instrum. Meas. 71 (2022) 1–11.
[19] V. Sesta, et al., Time-to-digital converters and histogram builders in SPAD arrays for pulsed-LiDAR, Measurement 212 (2023) 112705.
[20] K. Cui, et al., A high-linearity vernier time-to-digital converter on FPGAs with improved resolution using bidirectional-operating vernier delay lines, IEEE Trans. Instrum. Meas. 69 (8) (2020) 5941–5949.
[21] M. Abbas, et al., A 23ps resolution time-to-digital converter implemented on low-cost FPGA platform, in: 2015 International Symposium on Signals, Circuits and Systems, ISSCS, 2015, pp. 1–4.
[22] F. Arvani, et al., A reconfigurable 5-channel ring-oscillator-based TDC for direct time-of-flight 3D imaging, IEEE Trans. Circuits Syst. II 69 (5) (2022) 2408–2412.
[23] J. Wu, et al., A novel TDC scheme: Combinatorial gray code oscillator based TDC for low power and low resource usage applications, in: 2019 5th International Conference on Event-Based Control, Communication, and Signal Processing, EBCCSP, 2019, pp. 1–7.
[24] N. Lusardi, et al., From multiphase to novel single-phase multichannel shift-clock fast counter time-to-digital converter, IEEE Trans. Ind. Electron. (2023) 1–9.
[25] Z. Jaworski, Verilog HDL model based thermometer-to-binary encoder with bubble error correction, in: 2016 MIXDES - 23rd International Conference Mixed Design of Integrated Circuits and Systems, 2016, pp. 249–254.
[26] J.Y. Won, et al., Time-to-digital converter using a tuned-delay line evaluated in 28-, 40-, and 45-nm FPGAs, IEEE Trans. Instrum. Meas. 65 (2016) 1678–1689.
[27] J.Y. Won, et al., Dual-phase tapped-delay-line time-to-digital converter with on-the-fly calibration implemented in 40 nm FPGA, IEEE Trans. Biomed. Circuits Syst. 10 (1) (2016) 231–242.
[28] C. Foley, Characterizing metastability, in: Proceedings Second International Symposium on Advanced Research in Asynchronous Circuits and Systems, 1996, pp. 175–184.
[29] J. Horstmann, et al., Metastability behavior of CMOS ASIC flip-flops in theory and test, IEEE J. Solid-State Circuits 24 (1) (1989) 146–157.
[30] Y. Wang, et al., A nonlinearity minimization-oriented resource-saving time-to-digital converter implemented in a 28 nm xilinx FPGA, IEEE Trans. Nucl. Sci. 62 (5) (2015) 2003–2009.
[31] P. Carra, et al., Auto-calibrating TDC for an SoC-FPGA data acquisition system, IEEE Trans. Radiat. Plasma Med. Sci. 3 (5) (2019) 549–556.
[32] Y. Wang, et al., Multichannel time-to-digital converters with automatic calibration in Xilinx Zynq-7000 FPGA devices, IEEE Trans. Ind. Electron. 69 (9) (2022) 9634–9643.
[33] G. Rebala, et al., An Introduction to Machine Learning, Springer, 2019.
[34] N. Lusardi, et al., Quantization noise in non-homogeneous calibration table of a TCD implemented in FPGA, in: 2014 IEEE Nuclear Science Symposium and Medical Imaging Conference, NSS/MIC, 2014, pp. 1–5.
[35] F. Baronti, et al., On the differential nonlinearity of time-to-digital converters based on delay-locked-loop delay lines, IEEE Trans. Nucl. Sci. 48 (6) (2001) 2424–2431.
[36] J.-P. Jansson, et al., Enhancing nutt-based time-to-digital converter performance with internal systematic averaging, IEEE Trans. Instrum. Meas. 69 (6) (2020) 3928–3935.
[37] N. Lusardi, et al., 8-channels high-resolution TDC in FPGA, in: 2015 IEEE Nuclear Science Symposium and Medical Imaging Conference, NSS/MIC, 2015, pp. 1–2.
[38] K.-J. Choi, et al., Design and calibration techniques for a multichannel FPGA-based time-to-digital converter in an object positioning system, IEEE Trans. Instrum. Meas. 70 (2021) 1–9.
[39] J. Kalisz, et al., Field-programmable-gate-array-based time-to-digital converter with 200-ps resolution, IEEE Trans. Instrum. Meas. 46 (1) (1997) 51–55.

[40] R. Salomon, et al., BOUNCE: A new high-resolution time-interval measurement architecture, IEEE Embedded Syst. Lett. 1 (2009) 56–59.
[41] S. Tancock, et al., A 5.25ps-resolution TDC on FPGA using DSP blocks, 2019.
[42] C. Liu, et al., A 3.9 ps RMS resolution time-to-digital converter using dual-sampling method on kintex UltraScale FPGA, in: 2016 IEEE-NPSS Real Time Conference, RT, 2016, pp. 1–3.
[43] 7 series fpgas configurable logic block (ug474), 2016.
[44] UltraScale architecture configurable logic block(ug574), 2017.
[45] Z. Song, et al., A 256-channel, high throughput and precision time-to-digital converter with a decomposition encoding scheme in a Kintex-7 FPGA, J. Instrum. 13 (05) (2018) P05012.
[46] N. Lusardi, et al., Fully-migratable TDC architecture for FPGA devices, in: 2016 IEEE Nuclear Science Symposium, Medical Imaging Conference and Room-Temperature Semiconductor Detector Workshop, NSS/MIC/RTSD, 2016, pp. 1–3.
[47] J. Wu, et al., The 10-ps wave union TDC: Improving FPGA TDC resolution beyond its cell delay, in: 2008 IEEE Nuclear Science Symposium Conference Record, 2008, pp. 3440–3446.
[48] G. Cao, et al., An 18-ps TDC using timing adjustment and bin realignment methods in a Cyclone-IV FPGA, Rev. Sci. Instrum. 89 (5) (2018) 054707.
[49] E. Bayer, et al., A high-resolution (<10 ps RMS) 48-channel time-to-digital converter (TDC) implemented in a field programmable gate array (FPGA), IEEE Trans. Nucl. Sci. 58 (4) (2011) 1547–1552.
[50] L. Zhao, et al., The design of a 16-channel 15 ps TDC implemented in a 65 nm FPGA, IEEE Trans. Nucl. Sci. 60 (5) (2013) 3532–3536.
[51] M. Arredondo-Velázquez, et al., Trimmed-TDL-based TDC architecture for time-of-flight measurements tested on a cyclone V FPGA, IEEE Trans. Instrum. Meas. 72 (2023) 1–9.
[52] A. Tontini, et al., Design and characterization of a low-cost FPGA-based TDC, IEEE Trans. Nucl. Sci. 65 (2) (2018) 680–690.
[53] J.Y. Won, et al., Dual-phase tapped-delay-line time-to-digital converter with on-the-fly calibration implemented in 40 nm FPGA, IEEE Trans. Biomed. Circuits Syst. 10 (1) (2016) 231–242.
[54] M. Parsakordasiabi, et al., A low-resources TDC for multi-channel direct ToF readout based on a 28-nm FPGA, Sensors 21 (1) (2021).
[55] P. Kwiatkowski, et al., Bubble-proof algorithm for wave union TDCs, Electronics 11 (1) (2022).
[56] H. Chen, et al., Multichannel, low nonlinearity time-to-digital converters based on 20 and 28 nm FPGAs, IEEE Trans. Ind. Electron. 66 (4) (2019) 3265–3274.
[57] X. Yu, et al., A 4.8 ps root-mean-square resolution time-to-digital converter implemented in a 20 nm Cyclone-10 GX field-programmable gate array, Rev. Sci. Instrum. 93 (8) (2022) 085001.
[58] J. Kuang, et al., A 5.5 ps time-interval RMS precision time-to-digital convertor implemented in intel arria 10 FPGA, 2018, arXiv: Instrumentation and Detectors.
[59] N. Lusardi, et al., Very high-performance 24-channels time-to-digital converter in xilinx 20-nm kintex UltraScale FPGA, in: 2019 IEEE Nuclear Science Symposium and Medical Imaging Conference, NSS/MIC, 2019, pp. 1–4.
[60] J. Kim, et al., Linearity improvement of UltraScale+ FPGA-based time-to-digital converter, Nucl. Eng. Technol. 55 (2) (2023) 484–492.
[61] N. Lusardi, et al., Cross-talk issues in time measurements, IEEE Access 9 (2021) 129303–129318.
[62] F. Garzetti, et al., Time-to-digital converter IP-core for FPGA at state of the art, IEEE Access PP (2021) 1–1.
[63] J. Rivoir, Statistical linearity calibration of time-to-digital converters using a free-running ring oscillator, in: 2006 15th Asian Test Symposium, 2006, pp. 45–50.
[64] R. Szplet, et al., A 2.9 ps equivalent resolution interpolating time counter based on multiple independent coding lines, Meas. Sci. Technol. 24 (3) (2013) 035904.
[65] S. Berrima, et al., Fine resolution delay tuning method to improve the linearity of an unbalanced time-to-digital converter on a Xilinx FPGA, IET Circuits Devices Syst. 14 (2020) 1243–1252.
[66] J. Kuang, et al., A 128-channel high performance time-to-digital converter implemented in an UltraScale FPGA, in: 2017 IEEE Nuclear Science Symposium and Medical Imaging Conference, NSS/MIC, 2017, pp. 1–4.
[67] S. Bourdeauducq, A 26 ps RMS time-to-digital converter core for spartan-6 FPGAs, 2013, arXiv: Instrumentation and Detectors.
[68] W. Xie, et al., Efficient time-to-digital converters in 20 nm FPGAs with wave union methods, IEEE Trans. Ind. Electron. 69 (1) (2022) 1021–1031.
[69] N. Lusardi, et al., The role of sub-interpolation for delay-line time-to-digital converters in FPGA devices, Nucl. Instrum. Methods Phys. Res. A (2019).
[70] V. Gupta, et al., An overview of supervised machine learning algorithm, in: 2022 11th International Conference on System Modeling & Advancement in Research Trends, SMART, 2022, pp. 87–92.
[71] N. Dahiya, et al., A review paper on machine learning applications, advantages, and techniques, ECS Trans. 107 (1) (2022) 6137.
[72] T. Bartz-Beielstein, Supervised learning: Classification and regression, in: E. Bartz, T. Bartz-Beielstein (Eds.), Online Machine Learning: A Practical Guide with Examples in Python, Springer Nature Singapore, Singapore, 2024, pp. 13–22, http://dx.doi.org/10.1007/978-981-99-7007-0_2.
[73] M. Iqbal, et al., Supervised machine learning approaches: A survey, Int. J. Soft Comput. 5 (2015) 946–952.
[74] S.R. Dubey, et al., Activation functions in deep learning: A comprehensive survey and benchmark, Neurocomputing 503 (2022) 92–108.
[75] D. Choi, et al., On empirical comparisons of optimizers for deep learning, 2019, ArXiv arXiv:1910.05446.
[76] H. Chen, et al., Choice of activation function in convolutional neural networks for person re-identification in video surveillance systems, Program. Comput. Softw. 48 (5) (2022) 312–321.
[77] T. Szandała, Review and comparison of commonly used activation functions for deep neural networks, in: A.K. Bhoi, P.K. Mallick, C.-M. Liu, V.E. Balas (Eds.), Bio-Inspired Neurocomputing, Springer Singapore, Singapore, 2021, pp. 203–224, http://dx.doi.org/10.1007/978-981-15-5495-7_11.
[78] R. Szplet, et al., Measurement uncertainty of precise interpolating time counters, IEEE Trans. Instrum. Meas. 68 (11) (2019) 4348–4356.
[79] R. Szymanowski, et al., Quantization error in precision time counters, Meas. Sci. Technol. 26 (2015).
[80] J.-P. Jansson, et al., A CMOS time-to-digital converter with better than 10 ps single-shot precision, IEEE J. Solid-State Circuits 41 (6) (2006) 1286–1296.
[81] J.A. Nelder, et al., A simplex method for function minimization, Comput. J. 7 (1965) 308–313.
[82] J. Nocedal, et al., Numerical Optimization, 2e, Springer, New York, NY, USA, 2006.
[83] M.J.D. Powell, An efficient method for finding the minimum of a function of several variables without calculating derivatives, Comput. J. 7 (1964) 155–162.
[84] R. Storn, et al., Differential evolution - A simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (1997) 341–359.
[85] D.P. Kingma, et al., Adam: A method for stochastic optimization, 2014, CoRR arXiv:1412.6980.
[86] E. Okewu, et al., Parameter tuning using adaptive moment estimation in deep learning neural networks, in: O. Gervasi, B. Murgante, S. Misra, C. Garau, I. Blečić, D. Taniar, B.O. Apduhan, A.M.A.C. Rocha, E. Tarantino, C.M. Torre, Y. Karaca (Eds.), Computational Science and Its Applications, ICCSA 2020, Springer International Publishing, Cham, 2020, pp. 261–272.
[87] S. Ruder, An overview of gradient descent optimization algorithms, 2016, ArXiv arXiv:1609.04747.
[88] [Online]. Available: https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/.
[89] [Online]. Available: https://builtin.com/machine-learning/adam-optimization.
[90] [Online]. Available: https://www.kdnuggets.com/2022/12/tuning-adam-optimizer-parameters-pytorch.html.
[91] T. Yu, et al., Hyper-parameter optimization: A review of algorithms and applications, 2020, ArXiv arXiv:2003.05689.
[92] N. Corna, et al., Digital instrument for time measurements: Small, portable, high-performance, fully programmable, IEEE Access 9 (2021) 123964–123976.
[93] [Online]. Available: https://tediel.com/.
[94] M. Reyad, et al., A modified adam algorithm for deep neural network optimization, Neural Comput. Appl. 35 (23) (2023) 17095–17112.
[95] S. Natarajan, et al., Comparative analysis of different parameters used for optimization in the process of speaker and speech recognition using deep neural network, in: 2022 International Conference on Future Trends in Smart Communities, ICFTSC, 2022, pp. 12–17.
[96] C.-H. Chen, et al., A study of optimization in deep neural networks for regression, Electronics 12 (14) (2023).