POLITECNICO DI TORINO
Repository ISTITUZIONALE

A novel FPGA-based time-to-digital converter featuring machine learning-aided self-calibration

Original
A novel FPGA-based time-to-digital converter featuring machine learning-aided self-calibration / Amini Bardpareh, A., Vacca, E., Nicolini, D., De Sio, C., Azimi, S., Sterpone, L., Fiorina, E., Data, E.M., Mas Milian, F.. - In: INTELLIGENT SYSTEMS WITH APPLICATIONS. - ISSN 2667-3053. - 30:(2026). [10.1016/j.iswa.2026.200644]

Availability:
This version is available at: 11583/3008177 since: 2026-03-04T12:20:29Z

Publisher:
Elsevier

Published
DOI:10.1016/j.iswa.2026.200644

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

16 October 2026

Intelligent Systems with Applications 30 (2026) 200644

Contents lists available at ScienceDirect

Intelligent Systems with Applications

journal homepage: www.journals.elsevier.com/intelligent-systems-with-applications

A novel FPGA-based time-to-digital converter featuring machine learning-aided self-calibration

Arash Amini Bardpareh [a,*], Eleonora Vacca [a], Davide Nicolini [a], Corrado De Sio [a], Sarah Azimi [a], Luca Sterpone [a], Elisa Fiorina [b], Emanuele Maria Data [b], Felix Mas Milian [b]

[a] Dipartimento di Automatica e Informatica, Politecnico di Torino, Torino, Italy
[b] Istituto Nazionale di Fisica Nucleare, Torino, Italy

ARTICLE INFO

Keywords:
FPGA
Machine learning
Neural network
Time-to-digital converter

ABSTRACT

Time-to-digital converters (TDCs) implemented on Field-Programmable Gate Arrays (FPGAs) encounter significant challenges in achieving high precision. Unlike Application-Specific Integrated Circuits, FPGA-based TDCs must depend on standard logic elements as delay units, with each contributing different propagation delays due to process variations and available routing resources. These inconsistencies introduce non-uniformities in time measurement, degrading accuracy and thus requiring extensive calibration. A second major challenge is adapting analog-based time measurement techniques to the inherently digital nature of FPGAs. This work proposes a novel approach that eliminates the lengthy, time-consuming manual calibration by combining placement and routing optimization with machine learning techniques. We propose an approach based on custom placement and routing to minimize disturbance, while remaining errors due to process variation are compensated exploiting machine learning-based correction models. The work proposes and evaluates three different Machine-Learning (ML) models to interpret raw TDC outputs. Exploiting ML, we achieve a high-precision time measurement system capable of addressing the disturbances and non-linearities intrinsic to FPGA-based TDCs. This approach significantly reduces design complexity, accelerates deployment, and enhances the precision of FPGA-based TDCs, making them more scalable and suitable for a broad range of applications. Experimental results on a Kintex UltraScale FPGA show that the proposed ML-aided TDC achieves a timing precision below 15 ps, improving the precision of a conventional encoder-based FPGA TDC by more than 10.5 × and reaching performance comparable to a state-of-the-art ASIC TDC.

1. Introduction

Time-to-digital converters (TDCs) measure the time interval between two events by converting timing information into a digital value. They are widely used in applications such as LiDAR (Genschow, 2015), medical imaging (Garzetti et al., 2019), particle physics (Roy et al., 2017), and high-speed communication (Zhang et al., 2012), where high-resolution time measurement is critical. Traditionally, TDCs are implemented using Application-Specific Integrated Circuits (ASICs), which offer high precision and performance but with high development costs, low flexibility, and a long time-to-market. Recent advances in Field-Programmable Gate Arrays (FPGAs) have made them a practical and cost-effective alternative for implementing TDCs with custom characteristics (Fishburn et al., 2013; Lusardi et al., 2022).

Tapped Delay Line (TDL) is one of the most used architectures for the implementation of TDC on FPGA. TDLs rely on a series of logic elements, each introducing a small propagation delay (Bayer & Traxler, 2011). The first event triggers a signal traversing this series. When a second reference event happens, observing how much the signal propagated in the chain is possible to measure time between the two events with very fine resolution, much smaller than a single clock period. In FPGA-based TDCs, fabric resources are used as delay elements to build the delay lines, and technology scaling has reduced the delay introduced by each block to a value small enough to achieve high-resolution and

* Corresponding author.
E-mail addresses: arash.aminibardpareh@polito.it (A.A. Bardpareh), eleonora.vacca@polito.it (E. Vacca), davide.nicolini@polito.it (D. Nicolini), corrado.desio@polito.it (C. De Sio), sarah.azimi@polito.it (S. Azimi), luca.sterpone@polito.it (L. Sterpone), elisa.fiorina@to.infn.it (E. Fiorina), emanuelemaria.data@to.infn.it (E.M. Data), masmilia@to.infn.it (F.M. Milian).

https://doi.org/10.1016/j.iswa.2026.200644
Received 30 October 2025; Received in revised form 3 February 2026; Accepted 24 February 2026
Available online 25 February 2026
2667-3053/© 2026 The Authors. Published by Elsevier Ltd. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

performance comparable to that of ASIC-based TDC implementations. Modern FPGAs offer sufficient logic density with low propagation delays, enabling TDCs with picosecond-scale resolution based on tapped delay lines, while also offering the possibility to be tailored to the specific application.

However, despite FPGA advantages, significant challenges arise from process variability in both FPGA fabric logic and design implementations. Unlike ASICs, which can be designed to address precise time measurements, FPGAs are optimized for complex sequential digital designs rather than combinational designs such as accurate time measurement applications. Consequently, variations caused by the technological process, and additionally differences in design implementations, such as different routing path delays, clock jitters, and placement-dependent skew (Fishburn et al., 2013), create discrepancies in evaluating the delay introduced by the logic elements of the TDL. These factors can severely impact the linearity and resolution of the TDC, leading to degraded timing, accuracy, and resolution. A common manifestation of such issues is the presence of bubble errors in the thermometer code generated by delay-line-based TDCs (Liu, 2016). The thermometer code encodes the extent of signal propagation along the delay line, represented as a sequence of logic '1's and '0's. A bank of memory element samples the output of the delay elements. Such outputs are encoded as '1' if reached by the propagating signal, otherwise, they remain at '0'. In an ideal delay line, this sequence would consist of a continuous sequence of ones followed by a sequence of zeros, reflecting the ordered structure of the chain. When isolated zeros interrupt this monotonic sequence of ones, it is called *bubble error*. For instance, instead of a valid thermometer code like 11,110 in a delay line with 5 taps, the TDL might produce 11,010, where each bit indicates whether the signal has reached a given delay element (1) or not (0). Correcting and understanding this effect is non-trivial, as standard thermometer-to-binary encoders often filter out these anomalies by suppressing isolated ones or converting intermediate zeros into ones. Moreover, their dependence on several dynamic factors, including signal routing, the skew between the sampling and the propagating signal, and other sources of temporal misalignment within the FPGA fabric and operating parameters like temperature and voltage fluctuations make them difficult or impossible to eliminate. Typically, methods for processing variation-induced errors require manual intervention, necessitating extensive calibration. This process involves collecting and analyzing large datasets while continuously refining the encoder modules and the calibration procedure to correct timing discrepancies. Although this approach can ultimately achieve good accuracy by mitigating the various disturbances, it is inherently time-consuming, complex, and reliant on extensive domain expertise.

In many real-time sensing and monitoring domains, machine-learning models are increasingly used when the relationship between raw measurements and the desired output is complex and difficult to capture with fixed, hand-tuned rules. Examples include intelligent transportation, where short-term traffic forecasting benefits from data-driven models and real-time data integration, and IoT-based health monitoring, where fog/edge architectures are used to support low-latency screening and monitoring close to the data source (Fan et al., 2025; Jain et al., 2023; Khullar et al., 2022). In parallel, model visualization and interpretability techniques are widely studied to support understanding and debugging of deep-learning pipelines (Singh et al., 2022). Because these applications often operate under strict latency, power, and bandwidth constraints, deployment is frequently pushed toward embedded/edge hardware; in particular, low-precision (quantized/binarized) neural networks can be mapped efficiently to FPGAs to achieve low-latency inference with manageable compute and memory cost (Umuroglu et al., 2017). Motivated by these trends, we investigate whether a compact learned correction stage can replace iterative, hand-tuned decoding/calibration in FPGA-based TDCs while remaining practical for on-chip, real-time operation.

The main contribution of this work is a novel FPGA-based TDC that addresses the typical limitations of implementations in programmable logic, specifically targeting non-linearity, bubbles, and calibration overhead. The proposed TDC has been developed to meet the requirements of the HONEY (Hybrid ONline technology for particle therapy) research project, which focuses on real-time monitoring in Charged Particle Therapy (CPT) for cancer treatment. Our approach relies on hardware-optimized place-and-route and machine learning-driven self-calibration mechanisms, providing enhanced timing accuracy, reduced calibration complexity, and online post-processing by eliminating explicit, manual, and bin-by-bin calibration procedures during operation. At the hardware level, we focus on reducing non-uniformity in the delay contributions of the TDL by applying fine-grained layout and placement constraints. This involves explicitly guiding synthesis and routing tools to allocate specific logic and interconnect resources to equalize propagation delays across all delay elements. However, this strategy alone cannot entirely suppress non-linearity or eliminate bubble errors arising from residual process variation and signal misalignment. In addition, TDC behavior in FPGA fabric can drift with operating conditions such as temperature and supply-voltage changes, which are difficult to monitor and compensate using fixed decoding rules. To address these limitations, we propose a learning-based correction stage that replaces traditional thermometer decoding and bin-by-bin calibration. This method leverages the observation that timing deviations and bubble patterns tend to follow deterministic, device-specific distributions. Three different machine learning models are proposed and trained to recognize these patterns directly from raw thermometer codes, mapping them to corrected digital timestamps. A key feature of the proposed approach is using lightweight models, specifically designed to be easily trainable and efficiently deployable directly on the FPGA. While this paper evaluates the approach under nominal laboratory conditions at room temperature, the on-chip model also can be extended to multiple operating conditions (training condition-specific model parameters and selecting/reloading them based on measured temperature). We validate the proposed approach using a fully implemented TDC architecture deployed on a commercial FPGA. The performance and implementation costs are analyzed and discussed, highlighting the trade-offs between accuracy, complexity of the three models. The proposed approach is compared against a conventional encoder-based design implemented on the same hardware, as well as against a state-of-the-art ASIC TDC used as a reference.

The results demonstrate that combining layout-aware optimization and machine learning-based correction significantly improves timestamp reconstruction accuracy, achieving more than a 10.5 $\times$ precision improvement over the encoder-based solution. Difference in result suitable ML solution are discussed with a measurement standard deviation below 11 ps, the performance is comparable to that of ASIC implementations, eliminating the need for manual calibration, complex post-processing, and the high costs associated with ASIC design.

2. State of the art

TDCs in FPGA are predominantly based on a TDL architecture (Bayer & Traxler, 2011; Fishburn et al., 2013; Lusardi et al., 2022). TDL architectures leverage a chain of delay elements to create a fine-grained time quantization mechanism, where the propagation of a signal is sampled across multiple taps along the delay line. By capturing the state of each tap at a specific time, the TDC generates a thermometer code that reflects the signal's propagation delay, which can then be converted into a digital time measurement with high temporal resolution. However, process variations introduce inconsistencies in the delays of individual elements and routes, leading to non-uniform propagation times across the taps. These variations affect the linearity and resolution of the TDC, often requiring calibration techniques to compensate for uneven bin widths and to ensure accurate time measurement. The use of carry chains is particularly common in FPGA-based TDCs due to their short

and relatively consistent propagation delays compared to general-purpose logic. However, despite their advantages, carry chains still remain susceptible to placement constraints, routing irregularities, and process-induced variations that can impact timing uniformity. Managing these sources of disturbance is a significant challenge in FPGA-based TDC design. Unlike traditional ASIC TDCs, where decoding and calibration may be sufficient, FPGA implementations often require additional techniques and methodologies to model, limit, and compensate for these effects. State-of-the-art solution involves decoding and calibration, but additional techniques are required to improve TDC accuracy. In Lusardi et al. (2022), the authors propose to enhance TDC resolution by adopting the wave union technique coupled with the sub-interpolation method. Their approach uses multiple parallel TDLs to perform redundant time measurements, which are then merged into a single output via sub-interpolation. While this enhances resolution by a factor of N (the number of parallel TDLs), periodic bin-by-bin calibration remains necessary to compensate for process variations, which is an expensive methodology in terms of both resources and time, exacerbating further the calibration process. Another strategy for increasing resolution is presented in Liu (2016), where the authors propose a dual sampling, which applies two distinct sampling schemes to the same TDL. Multiple registers capture the state of the delay elements at different points in time, effectively creating overlapping samples. Specifically, in Liu (2016), the dual sampling is achieved by exploiting the two available propagation paths of the two registers associated with the FPGA carry chain blocks. The dual sampling is implemented in such a way that in the primary propagation path, the signal propagation is captured as a transition from 0 to 1 in the register state, while in the second path, it is the opposite. Having two transitions captured for each delay element has proven to effectively increase the TDL resolution.

Various post-processing techniques also exist to refine the accuracy of TDCs beyond TDL-level enhancements. Calibration methods such as bin-by-bin calibration and offset calibration compensate for variations due to systematic process, voltage, and temperature (PVT) fluctuations. A common technique to mitigate non-linearity in bins is bin decimation (Castelvero et al., 2024), where bins exhibiting non-uniform delays are merged to improve overall linearity at the cost of slightly reduced resolution. Beyond these TDL-level improvements, recent FPGA-based TDC designs increasingly combine multiple enhancement techniques to improve resolution and linearity while keeping calibration manageable. For example, recent UltraScale/UltraScale+ implementations report high-resolution operation by combining wave-union concepts with dual-sampling, sub-structures, and efficient encoders (Castelvero et al., 2024). Other work focuses on improving linearity through delay-line restructuring and histogram-based correction methods, including decimated delay-line concepts and multi-chain approaches (Kim et al., 2023) In parallel, alternative encoding strategies have been proposed to reduce empty bins and improve DNL/INL behavior while maintaining flexibility (Hua & Chitnis, 2022). Finally, large multi-channel high-throughput FPGA-TDC systems have been demonstrated using online calibration together with robust encoding techniques to sustain high event rates (Liu & Wang, 2015). In addition to these hardware- and calibration-centric techniques, promising results have recently been achieved by Garzetti et al. (2024) exploiting the machine learning algorithm in the post processing phase to account for PVT errors. The authors in Garzetti et al. (2024) proposed to rely on a python-based multilayer perceptron, composed of 10 fully connected layers, to perform the matching between digital representation and timestamps. Although the approach achieved high precision, it operates offline on a standalone PC, limiting its suitability for real-time applications.

In contrast to existing approaches that compensate for non-linearity and process variation through super sampling strategies, such as multi-TDL architectures or dual-edge sampling, and rely heavily on comprehensive calibration, our method introduces a fundamentally different mechanism: TDL architecture enhanced by placement-aware optimization and featuring a lightweight on-chip machine learning-based decoding and correction module, fully integrated within TDC design and working online in real-time. This approach is particularly relevant in our application context, and in general in scenarios where scalability to many parallel TDC channels is essential for receiving and correlating signals from multiple sensors. Recent machine-learning-based solutions like (Garzetti et al., 2024) are based on 10 fully connected layers trained to perform thermometer decoding and calibration offline with a structure similar to the original encoder. While effective regarding results, this architecture is too large and computationally intensive for practical deployment inside an FPGA without dedicated ML acceleration hardware. In contrast, our model is intentionally lightweight, typically consisting of two small dense layers with fewer neurons, enabling direct deployment on a commercial FPGA. To further enhance runtime performance, we implement a systolic array structure to accelerate the evaluation of the ML model. While the design we propose operates with single channel, the architecture is designed with parallel scalability in mind. The lightweight nature of the model and the systolic structure ensure that multiple TDC channels can be supported in parallel without introducing significant resource overhead, replicating the TDLs. Moreover, the model can be trained on a modest dataset collected directly from the deployed TDC hardware, avoiding the need for large-scale data generation or offline training infrastructure. Once trained, it can run entirely on-chip in real time, with no need for external computation or post-processing.

3. Application context: honey research project

The proposed TDC has been developed to meet the specific functional requirements of the HONEY research project, which aims to enhance the precision and effectiveness of CPT for cancer treatment (Ranjbar, 2026). CPT uses protons and carbon ions to target tumors while minimizing damage to surrounding healthy tissues, requiring precise monitoring of beam parameters and particle range. A critical part of optimizing CPT is the ability to monitor the delivered dose and beam range in real time, which requires accurate detection and characterization of the particles generated during irradiation. The project uses in-beam Positron Emission Tomography (PET), which tracks positron emitters within the patient by detecting annihilation photons. As primary particles traverse the patient's body and interact with tissue, they deposit energy and generate secondary particles, which are tracked relying on detectors with high time resolution. Measuring the Time-of-Flight (ToF) (Villa et al., 2013) and energy of these secondary particles provides essential information for verifying treatment efficiency and tumor characteristics. However, challenges for enhancing PET for CPT treatment verification include low statistics and positron emitter wash-out. The measurements must be performed under the constraints of limited acquisition time, high event rates, multiple channels, and high timing accuracy, especially during the short time windows available in clinical CPT sessions. To address such issues, the HONEY project proposes a multi-detector system with compact devices that can be placed near the patient to collect multi-channel data during irradiation. This data is processed through the TDC implemented on the FPGA, which must handle multiple data channels simultaneously. The advantage of using an FPGA lies in its flexibility, which allows for the integration of custom logic close to the detector front-end, with a high level of parallelization and customization. Although this makes it a unvaluable resource for CPT treatment verification, it adds complexity to the TDC implementation.

In the initial phase of the project, the setup relied on the picoTDC module (Altruda et al., 2023) developed at CERN, which served as a reliable reference for picosecond-level timing measurements. However, being an ASIC, the picoTDC offers limited adaptability as the system evolves, particularly regarding data acquisition setup, online analysis of data, and the use of a multichannel architecture. To greater flexibility, scalability, and tighter integration with the detector electronics, a custom FPGA-based TDC was identified as suitable solution to replace

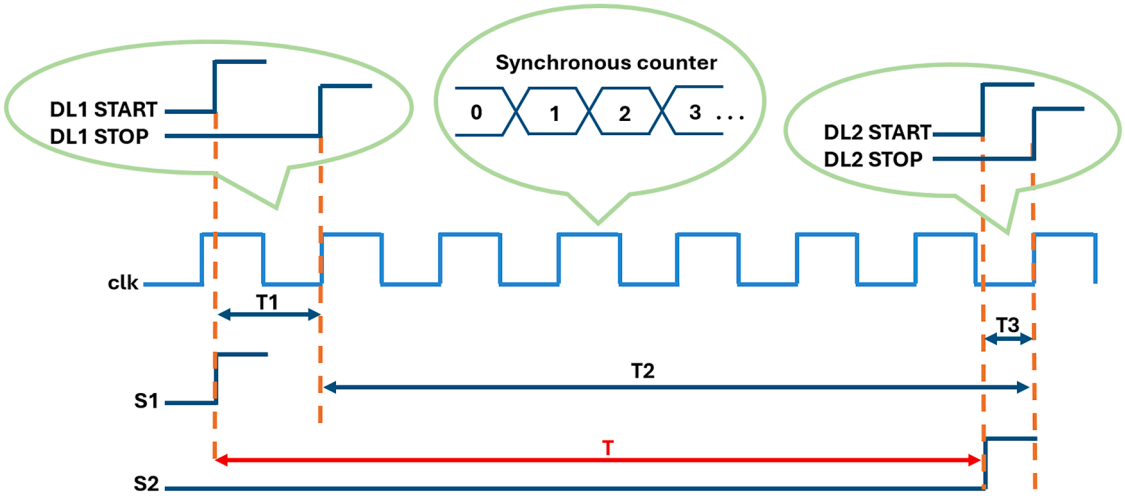

Fig. 1. The measurement principle of the TDL architecture.

the picoTDC in the HONEY setup.

In particular, fulfilling the requirements of in-beam PET for CPT introduces a set of stringent requirements. The project calls for multi-channel, real-time processing of time-resolved data during patient irradiation, where each detection channel must timestamp events with precision approaching the tens of picoseconds, sustain high event rates, and remain synchronized across the full acquisition window. These conditions push the limits of what is practically achievable on FPGAs, which are inherently optimized for digital logic, not high-precision timing or analog-like delay uniformity. FPGAs suffer from significant process variability and routing unpredictability, which introduces timing skews between elements that should be identical. These effects particularly affect delay-line-based architectures, where precise propagation delay uniformity is essential. In this context, achieving high timing resolution and preparing for multi-channel scalability introduces several non-trivial challenges. While the current implementation proposed in this work focuses on a single TDC channel, the system must be designed with future expansion in mind, where multiple detectors will need to operate in parallel, and each signal must be independently timestamped. This scenario poses significant constraints on calibration and architectural scalability.

While ideal for flexible digital logic and parallelism, FPGAs present significant obstacles when scaling delay-line-based TDCs across multiple channels. Each additional channel increases pressure on placement, routing, and clock distribution resources, making it increasingly difficult to preserve uniform delay characteristics and deterministic timing across the system. These effects are exacerbated by the intrinsic process variation of FPGA logic elements and routing paths, which cause bin non-uniformities and skew that are difficult to correct at scale. Furthermore, calibration becomes increasingly complex in a multi-channel setting. Even in single-channel systems, fine-grained timing resolution requires careful tuning to mitigate non-linearity and ensure bin uniformity. Extending this to multiple TDLs implies performing individualized calibration per channel, and ensuring cross-channel consistency is crucial for applications like in-beam PET, where signals must be correlated across spatially separated detectors. Achieving this level of synchronization and stability without external calibration hardware or excessive reconfiguration overhead represents a significant technical barrier.

Although only one TDC channel has been implemented in the current design, the architecture has been consciously structured to accommodate future scaling, considering the calibration pipeline, parallel processing, logic reuse, and timing closure strategies. This design approach is intended to facilitate future scalability studies within the HONEY project, while acknowledging that multi-channel implementations will require dedicated validation and refinement in subsequent project stages.

4. TDC design and implementation

The proposed FPGA-based TDC design aims to meet the project requirements by achieving a resolution in the order of tens of picoseconds, with a measurable dynamic range of approximately 1 μs. The TDC must determine the time interval, denoted as T, between the occurrence of two input signals, S1 and S2, as shown in Fig. 1, which are generated by the sensor front-end for measuring the particle ToF.

We adopted a dual TDL-based architecture. In a Delay-Line (DL), a START signal is propagated through a chain of cascaded delay elements called taps. The delay introduced by this chain is used for estimating the time interval between the arrival of the START and STOP signals. Specifically, when the START signal is received, the reference edge begins traversing the delay line. When the STOP signal is received, the state of all taps is sampled to produce a thermometer code encoding the fine time measurement.

Typically, the dynamic range must be large enough to avoid operational constraints. However, simply extending the delay line to cover several nanoseconds would require an excessive number of resources. To overcome this, the TDL is coupled with a synchronous counter that performs coarse-grained measurement with clock-period resolution. In our implementation, the counter is combined with two independent delay lines. One line measures the time from the S1 signal to the next rising edge of the clock, while the other captures the same for the S2, referred to as T1 and T3, respectively. Concurrently, the counter tracks the clock cycles between these two edges (interval T2). Fig. 1 reports a visual representation of T1, T2, and T3 based on the S1 and S2 time of arrival and also highlights the relation of these signals with the START and STOP signal of each DL. The final time interval, named T, is computed using the expression (1). It is important to note that for correct operation, each delay line must be long enough to span at least one full clock cycle. Consequently, the length of the delay chain becomes a limiting factor that directly affects the maximum usable clock frequency.

$$T = T_1 + T_2 - T_3 \tag{1}$$

The choice of using two DL is due to various considerations. Measuring both T1 and T3 with a single delay line would require multiplexing the delay line input between the S1 and S2 signals and capturing the delay line state twice per measurement. However, this approach is not viable in high-precision scenarios for several reasons. First, multiplexing introduces asymmetric delays that degrade both timing resolution and measurement accuracy. Minimizing the delay

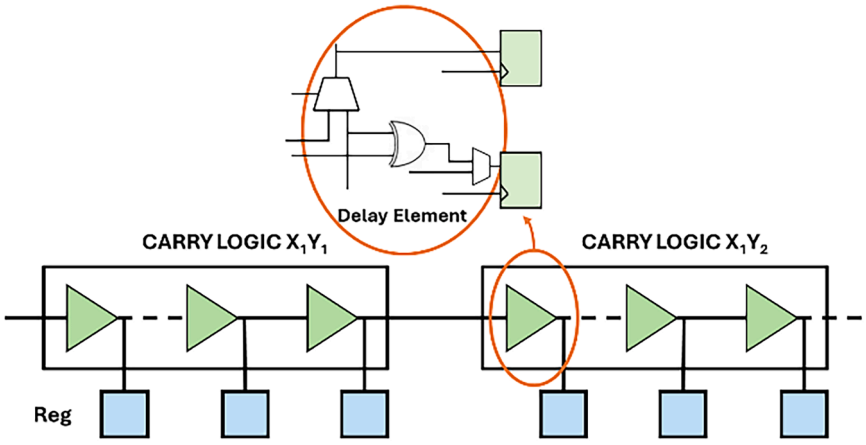

Fig. 2. Delay line using carry logic implemented on FPGA.

between the signal edge and the sampling operation on FPGAs is critical, as any additional logic along this path can compromise measurement fidelity. For the target application, signals must be captured and sampled as close as possible to their arrival point to preserve their precise timing characteristics. Indeed, even minimal multiplexing or gating logic on the delay-line input may introduce propagation delays on the order of several tens of picoseconds, which is comparable to or larger than the delay of a group single carry-chain tap. Although this value represents an order of magnitude and may vary depending on process variations, FPGA architecture, and fabrication technology, such additional delay would directly degrade the effective time resolution. Additionally, S2 happens within the same clock window of S1, the single delay line would not be available. For these reasons, the architecture employs two distinct delay lines, one for each input signal. In ideal TDLs, the effective resolution, typically called the Least Significant Bit (LSB) (Castelvero et al., 2024; Fishburn et al., 2013; Garzetti et al., 2024; Lusardi et al., 2022), is determined by the constant delay introduced by adjacent taps. This value defines the finest time increment the TDL can detect. On an FPGA, the LSB is not a configurable parameter but a physical property resulting from characteristics of the logic elements and routing available. To minimize LSB, the carry-chain elements are used as shown in Fig. 2, which are typically the fastest available FPGA resources, offering the shortest and most consistent propagation delay between adjacent taps.

Ideally, each delay element within the delay line is expected to introduce a precise and uniform amount of delay. If we consider the maximum measurable time by the TDL as T_{max} and assume there are N delay elements, the delay introduced by each element (LSB) can be calculated using expression (2).

$$T_{LSB} = \frac{T_{max}}{N} \tag{2}$$

Indeed, as the START signal propagates sequentially through the taps, the state of the taps in the delay line transitions linearly from a series of 0 s to a series of 1 s. At the moment, the STOP signal triggers the capturing registers, and the state of the delay line is recorded, forming a binary sequence known as thermometric code. In thermometric coding, the initial portion of the delay line contains ones (representing elapsed propagation), followed immediately by zeros (representing delays yet to be traversed). As propagation progresses through the delay elements, more zeros turn into ones. The resulting thermometric code directly corresponds to the elapsed time, ideally covering a range from zero up to at least one clock cycle period (t_{clk}) in TDL architecture, combined with the clock counter. If the thermometric code contains N number of ones

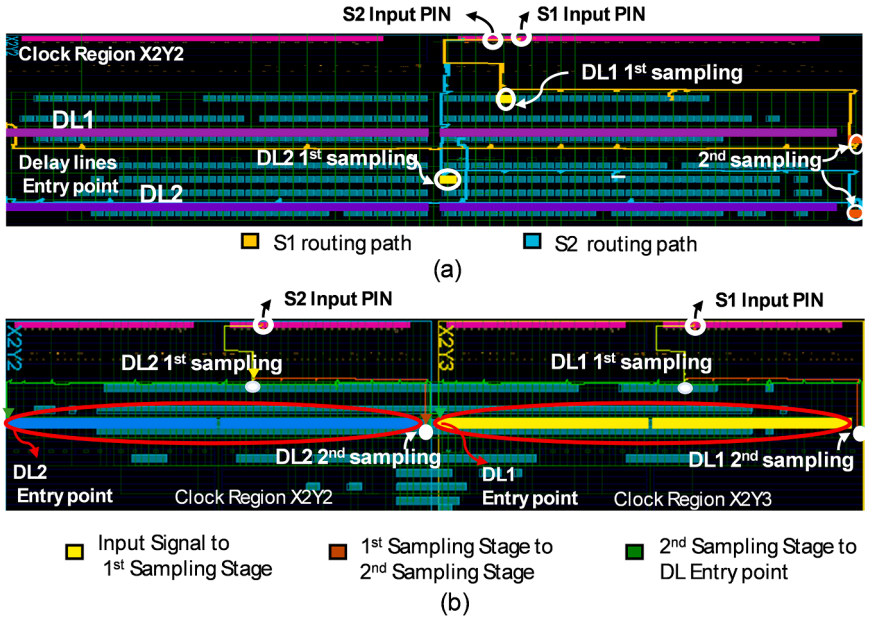

Fig. 3. Proposed TDC post implementation. In (a), the plain version, while in (b), the constrained layout.

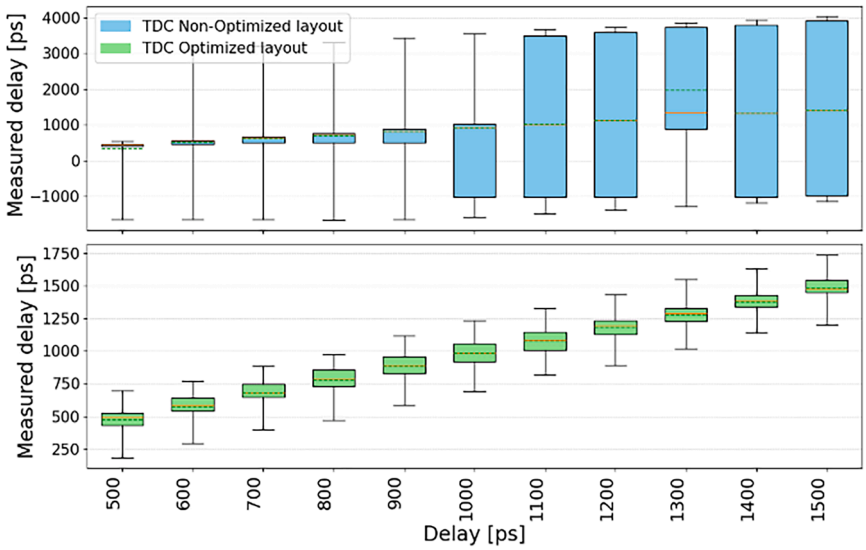

Fig. 4. Initial TDC with encoder Input–Output characteristics with and without synchronization between DLs and counter.

(1′s), the measured time interval can be determined using expression (3).

$$t = N * \ T_{LSB} \tag{3}$$

However, the delays introduced by individual taps are not uniform. In particular, some delay elements exhibit significantly higher delays than others, leading to ultra-wide bin, breaking linearity and degrading accuracy. The hierarchical and heterogeneous structure of the FPGA fabric primarily causes this non-uniformity. Within a single logic tile, propagation delay through carry-chain elements remains relatively consistent. However, as the delay line extends beyond the boundaries of a single tile, the signal must traverse longer interconnections that introduce additional and less predictable delays, often involving interconnection matrices and configurable routing. These routing paths frequently differ in physical length and resource contention, contributing to tap-to-tap delay variations. An even more significant contribution to delay irregularity also derives from clock regions. When the delay line crosses clock region boundaries, changes in routing structure and timing resources produce substantial shifts in the propagation delay. Such an issue is intrinsic to the physical architecture of FPGAs, making it challenging to achieve a perfectly uniform timing response.

To mitigate the impact of delay non-uniformity and maximize the linearity of the TDC response, we implemented a custom placement strategy. The default placement and routing performed by Computer-Aided-Design (CAD) tools for FPGA are typically optimized for digital logic and may not ensure consistent delay characteristics across long cascaded paths like those in a TDL. As a result, automatic placement often introduces irregularities that degrade the effective time resolution and increase bubble error probability and non-uniform bins for a TDL design, as shown in Fig. 3, where the post-implementation view of the two delay lines is proposed. As illustrated, the connection paths from the input pins to DL's entry point (S1 to DL1 in light orange and S2 to DL2 in light blue) differ significantly. These differences introduce a measurement offset between the two delay lines, which is further influenced by their physical placement on the FPGA (Xilinx, 2023; Zhang et, al.,n.d.). For example, inconsistencies in routing length between the S1 path to DL1 and the S1 path to the counter (and likewise for S2) may cause the counter to register an incorrect number of clock cycles, either counting many cycles more or too few. As a result, leaving the TDC layout entirely to the toolchain's discretion leads to unpredictable behavior that can deviate significantly from RTL simulations. Therefore, applying physical design constraints becomes essential to achieving reliable and consistent performance. The proposed placement strategy is implemented through a combination of manual floorplanning during an initial exploration phase and scripted placement and routing constraints. The exploratory phase is used to identify a symmetric and timing-efficient layout, while the scripted constraints are applied by the FPGA toolchain to enforce this layout in a repeatable and deterministic manner across builds, as well as to support incremental design modifications.

The optimal layout solution for the HONEY project specifications has been found in the highly symmetric implementation proposed in Fig. 3.

Typically, FPGAs adopt a regular two-dimensional grid of tiles, where logic resources such as CLBs and routing elements are arranged in rows and columns and grouped into clock regions. Each clock region is driven by a dedicated segment of the global clock network. In our implementation, particular care was taken to prevent the TDL from spanning clock-region boundaries. Although the global clock network provides very low skew (Xilinx, 2023), the residual skew is non-negligible for picosecond-scale measurements. As the minimum skew is achieved when routing remains within the same column of resources inside a single clock region, each delay line was placed entirely within one clock region to minimize skew-induced timing error. Additionally, interconnection matrices were completely avoided for linking delay elements; instead, we exploited intra-tile carry chains to propagate the signal. This strategy kept the delay elements confined to adjacent slices of the same column whenever possible, with single-column carry chains providing the most uniform and minimal propagation delay.

Although these manual placement approaches are highly demanding, requiring explicit manual assignment of all logic elements of the netlist, such as cells and nets, to the specific resources available on the FPGA, they significantly reduce non-linearity in the delay profile and limit the variation in LSB across the entire chain. The impact of the layout solution on DL behavior is shown in Fig. 4. Here, we show a comparison of the Input/Output (I/O) characteristics of the implemented TDC, working with a 400 MHz clock, with and without our custom layout optimization. On the x-axis is the input delay to be measured, i.e., the distance between the S1 and S2 time of arrival. On the y-axis, the measurement produced by the TDC over multiple runs for the same input delay. As illustrated, the I/O characteristic of the TDL without the custom layout optimization exhibits a significant error, up to 2500 ps in the worst cases, nearly equivalent to an entire system-clock period. This time misalignment effect arises from a timing mismatch between the delay line and the counter. In particular, when the input signal occurs very close to the next rising clock edge, the delay line and the counter do not register the event simultaneously. As a result, the counter output may be latched either earlier or later than intended. These large variations can be mitigated by selecting an optimal place-

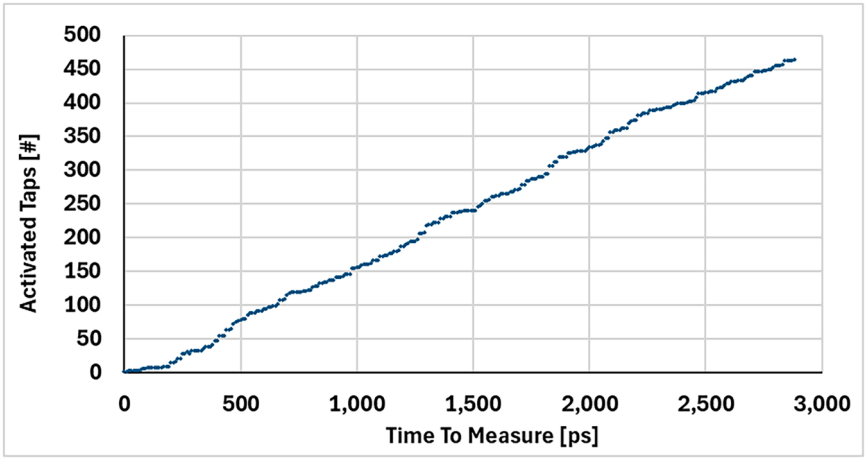

Fig. 5. Delay line using Carry logic implemented on FPGA input output behavior.

and-route solution, which yields a more stable and reliable TDC output and reduces the burden of subsequent calibration. Nevertheless, calibration remains essential due to residual non-idealities such as bubble errors in the thermometer code. These errors manifest as isolated zeros within the otherwise monotonic sequence of ones, disrupting the ideal propagation pattern of the delay line. Traditional thermometer-to-binary encoding schemes are generally ineffective at eliminating such artifacts and may even introduce additional errors when implemented on FPGA fabrics. Exhaustive manual calibration approaches, while theoretically capable of compensating for these irregularities, are impractical in real-world applications due to their complexity and limited scalability. Therefore, a robust strategy is required that combines hardware-level optimization, such as the improved place-and-route we proposed, with systematic, data-driven calibration techniques to correct delay irregularities and account for process variations.

Therefore, we integrated a learning-based calibration approach that complements the hardware-level optimizations described previously. Instead of relying on decoding rules or demanding manual tuning, we adopt a Neural Network (NN) approach that learns to map the raw, potentially disturbed thermometer codes directly to calibrated time values. This method corrects systematic deviations, including those introduced by non-uniform delay elements, signal asymmetries, and bubble errors. The neural network effectively functions as a decoder, trained to interpret the sampled state of the delay line and reconstruct the corresponding fine time interval. By learning from a representative dataset of known input delays, it captures complex non-linearities and compensates for physical inconsistencies across the TDC architecture. Most importantly, once deployed on the device, the trained model enables real-time timestamp computation directly within the FPGA, eliminating the need for post-processing on external systems and reducing latency in real-time applications.

5. Machine learning-aided self calibration

As discussed in the previous section, implementing a TDC on an FPGA is a challenging task, primarily due to the need to tackling effectively inherent non-linearities introduced by process variations and clock skew. Even when relying on a custom place-and-route strategy to mitigate these non-idealities, residual non-linearities persist, directly affecting the output of the TDLs. Consequently, the TDLs do not generate a perfect thermometer code that can be directly converted into a binary value. Fig. 5 illustrates the measured linearity of the TDC, highlighting how, despite an overall approximately linear behavior, noticeable fluctuations occur in the number of activated taps across the dynamic range. These irregularities result in uneven bin widths, undermining the precision of time measurements. Since achieving picosecond-level resolution requires a high degree of linearity, addressing these distortions is essential.

A common method for encoding the non-ideal thermometer code into a binary representation relies on sum-1 s decoders. These decoders sum up the active bits in the thermometer code, often applying bin-specific weighting to partially account for non-uniformities. However, in addition to deterministic errors, which can be mitigated through an extensive and time-consuming calibration procedure, sum-1 s decoding is also affected by statistical errors arising from metastability phenomena. These metastability-induced errors are inherently unpredictable and therefore cannot be systematically corrected, especially at operating time.

To address these limitations, we propose a fully integrated approach that removes the need for explicit decoding, statistical correction, or complex calibration stages. Multiple neural network architectures, including multilayer perceptron (MLP), deep neural network (DNN), and convolutional neural network (CNN) have been investigated and trained to process the inputs: namely, the raw thermometer codes from both delay lines together with the coarse counter value, and to produce the final timestamp as output. CNNs are particularly well suited to this task because thermometer codes exhibit strong local spatial correlations and structured transition patterns, which can be effectively captured by convolutional filters. A key advantage of our method is the use of a single, compact hardware accelerator capable of running all these models without any modification to either the TDC design or the computational logic. This reconfigurable setup makes it particularly convenient to experiment with or deploy networks of different depth and complexity while preserving identical I/O characteristics. By bypassing traditional decoding and correction techniques, the architecture remains robust, flexible, and efficient for high-resolution time measurements in FPGA-based TDCs.

A. The Neural Network Models

Neural networks offer an effective means of correcting the non-linearity inherent in FPGA-based TDCs, as they can directly learn the inverse mapping of the TDC's non-ideal transfer function, and generalize and understand different disturb and their correlation. Unlike conventional calibration methods, which rely on manual adjustment or exhaustive lookup tables, NNs naturally adapt to complex non-linear relationships introduced by delay irregularities, routing asymmetries, or metastability. Once trained, they are capable of inferring corrected timestamps directly from raw TDC outputs in real time, providing both accuracy and low-latency operation.

To explore this capability, we implemented three distinct NN models: (i) a lightweight fully connected network with a single hidden layer of

Table 1
Ablation study of the single-layer model under varying neuron counts.

# Neurons	SAE2 [ps]
16	8.86695
32	8.72664
64	8.67540
128	8.87242
256	9.14073

Table 2
Ablation study of the deep model under varying depth and width.

Hidden Layers #	128 + 64 neuron layers	64 + 32 neuron layers
3	9.04058	8.91496
4	8.8521	9.26023
5	8.69741	8.70114
6	8.85032	8.51955
7	9.14519	8.7786

64 neurons, optimized for low-latency inference on resource-constrained FPGA fabric; (ii) a deeper fully connected network with six layers (made of 128–128–64–64–64–1 neurons), designed to increase representational capacity and evaluate the trade-off between model complexity and correction accuracy; and (iii) a convolutional neural network which leverages local receptive fields to exploit spatial correlations in the 940-bit input vector (made of 12 bits from the coarse counter and 464 bits from each delay line output). The network consists of two one-dimensional convolutional blocks: the first applies 64 kernels of size 7 with stride 3, followed by a max-pooling layer of size 2 and stride 2; the second applies 32 kernels of size 6 with stride 2, followed by a 2-stride max-pooling layer. The resulting feature maps are flattened and fed into a fully connected layer with 128 neurons, followed by a single-neuron output layer. All hidden and convolutional layers use ReLU activations, with a single linear output neuron producing the corrected timestamp. All models were trained for 20 epochs using Huber loss, learning rate of 5e-5, batch size of 64, and Adam optimizer. It was also employed a decays of learning rate every 5 steps with gamma = 0.5.

To identify suitable model sizes and avoid over-parameterization, we conducted an ablation study for each neural network architecture by varying the number of neurons and, where applicable, the network depth. Tables 1 and 2 summarizes the results of this study for all models.

For the single-hidden-layer network, we evaluated configurations ranging from 16 to 256 neurons. The results show that smaller networks generally achieve better correction performance, with the best results obtained using 64 neurons.

For the deep fully connected model, the ablation study was designed to investigate both the optimal number of hidden layers and the

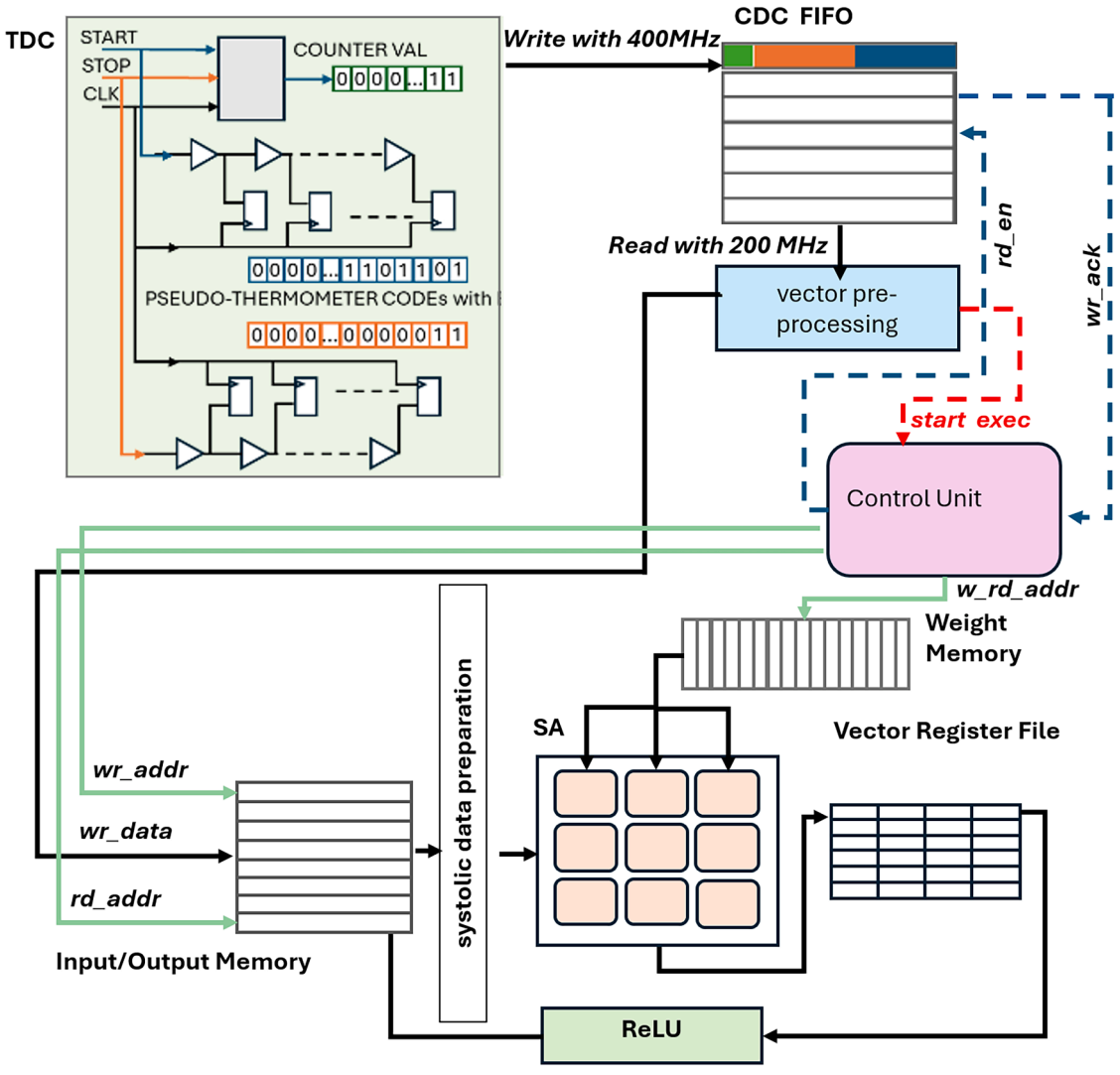

Fig. 6. Conceptual Schema of the proposed platform including TDLs, Systolic Array and CDC FIFO.

appropriate layer width. The network architecture was constrained to include three fixed hidden layers: two initial hidden layers at the input side and one hidden layer immediately preceding the output. Additional hidden layers were progressively inserted between these fixed layers. The two initial hidden layers were configured with twice the number of neurons used in the intermediate hidden layers. We evaluated architectures that added hidden layers of 64 and 32 neurons, with the first two layers containing twice the number of neurons of the subsequent layers. The results reported in Table 2 highlight the trade-off between model complexity and correction performance and guided the selection of the final architecture. The study indicates that a configuration with five hidden layers provides the best balance between accuracy and complexity, while further increasing the number of layers yields no statistically significant performance improvement

To accelerate inference, we integrated a lightweight systolic array (SA) as a hardware accelerator within the FPGA. The SA provides highly parallel execution of NN operations, enabling low-latency inference without relying on external processors or memory transfers. This parallelism allows the design to handle multiple input vectors simultaneously, supporting either multi-channel TDC systems or high-rate acquisitions on a single channel, which reflects realistic use cases such as time-correlated multi-detector experiments.

Beyond raw performance, the SA architecture also introduces a crucial advantage in flexibility and rapid prototyping. Different NN architectures, with varying levels of complexity and accuracy, can be deployed simply by modifying the assembly code and weight data loaded onto the accelerator. This removes the need to redesign the FPGA accelerator for each neural network variant, thereby greatly speeding up both design-space exploration and system prototyping. Neural models can be trained and tested offline, and then quickly deployed to the FPGA by updating only the weights, rather than the hardware structure.

This feature is also valuable even if aging effects or radiation-induced degradation may gradually alter the TDC characteristics, introducing additional discrepancies over time. In such scenarios, the NN can be retrained offline to compensate for the newly observed non-linearities, and the updated model can be deployed on the FPGA by uploading new weights, with no hardware redesign required. Thus, the combined use of neural networks and a systolic-array-based accelerator ensures not only efficient and accurate correction of TDC non-linearity, but also adaptability to evolving device conditions, enhancing both reliability and maintainability. Moreover, executing inference entirely within the FPGA opens the opportunity to implement additional stages of on-chip processing or refinement, such as filtering, classification, or compression.

We also considered classical regression models such linear regression, decision trees, or Support Vector Regression (SVR) as potential alternatives to neural-network-based solutions. However, the regression task addressed in this work operates on a high-dimensional binary input with significant non-linear disturbs of input–output. Under these conditions, linear regression models are generally insufficient to capture the required non-linear behavior. Other classical models, such as decision trees or SVR, typically require a large number of nodes or support vectors to achieve comparable accuracy on such high-dimensional data. As a result, their hardware implementation on FPGA is not necessarily lighter or easier. Tree- or forest-based models translate into large comparator networks and complex control logic. Moreover, the very high dimensionality of the input vector poses additional challenges for LUT-based implementations, given the limited fan-in of such logic elements commonly available on modern FPGAs. Discretizing such a large input space would be impractical without aggressive feature compression, which would in turn risk degrading timing accuracy. For these reasons, we focused on compact neural network models that can directly operate on raw delay-line codes without explicit feature extraction and that efficiently map onto the proposed systolic-array accelerator.

B. Systolic-Array for Machine Learning Acceleration

A general overview of the proposed architecture integrating the TDC with the SA accelerator is illustrated in Fig. 6. In our implementation, we adopted tinyTPU (Fuhrmann, 2018), an open-source Tensor Processing Unit (TPU) featuring its own instruction set architecture (ISA). The selection of this architecture is primarily motivated by its autonomous computational capability. Once the NN weights are appropriately loaded into memory and the model is converted and stored in the instruction memory, the accelerator executes the computations without requiring user intervention and directly produces the final output. Consequently, deploying a neural network on tinyTPU primarily involves converting the model into a format compatible with the accelerator, while also considering operation tiling within the weight matrix.

Since tinyTPU is intended to function as a co-processor, we modified the system to adapt to our needs. Specifically, we replaced the original instruction FIFO with a standard memory structure preloaded with the instructions corresponding to the implemented model. In the nominal functioning, instructions retrieved from the FIFO are discarded after execution, requiring them to be reloaded for each inference. However, in our implementation, tinyTPU operates in conjunction with the TDC and is expected to execute the same inference across consecutive measurements. To eliminate unnecessary reloading overhead and minimize computational latency, we opted for a persistent instruction memory approach. Another critical consideration in our design is the disparity in operating frequencies between components.

While tinyTPU, as a standalone implementation, can operate at a maximum frequency of 200 MHz, the TDC counter must function at a significantly higher frequency, potentially twice that of the TPU, to ensure precise time measurements. To accommodate this frequency difference, we implemented a multi-clock domain architecture, where the only interface between the TDC and tinyTPU is the transfer of the TDC's output vector as an input to the TPU. This data exchange is managed via a Clock Domain Crossing (CDC) FIFO (AMD, FIFO Generator v13.2 Product Guide (PG057), Document ID: PG057 2017), facilitating high-frequency data writing while allowing a lower-frequency data reading. Targeting AMD FPGA devices, the FIFO is instantiated in the design using AMD's FIFO Generator IP, configured with a port width of 1024 bits to accommodate the 940-bit output vector of the TDCA FIFO depth of 2048 entries was selected to decouple high-rate data acquisition from accelerator inference. According to the HONEY project specifications, the TDC can generate a new raw output vector every 1 μs during active acquisition phases. At this acquisition rate, a 2048-entry FIFO can buffer up to 2048 output vectors, corresponding to approximately 2 ms of continuos raw measurements. Importantly, in the target experimental context particle arrivals are typically organized in bursts (spills), characterized by short time windows with high event rates followed by intervals with little or no activity. In this scenario, buffering is required only to temporarily accumulate groups of measurements generated during active spill periods, while inference can proceed during subsequent idle intervals. Therefore, the FIFO depth determines the maximum burst duration that can be absorbed before overflow, rather than enforcing continuous real-time processing at the peak acquisition rate. The FIFO depth adopted in the proposal is intentionally conservative. In practice, the required buffer length can be parametrically dimensioned based on the expected number of events per spill and the specific acquisition profile of the target application, and is expected to be smaller for the considered use case.

For clarity, Fig. 6 highlights the two clock domains and the corresponding CDC boundary. The TDC logic, including delay-line sampling and coarse counting, operates in the TDC clock domain, while preprocessing, I/O buffering, and systolic-array computation operate in the accelerator clock domain. The CDC FIFO represents the only clock-domain crossing in the design and employs independent write and read clocks. FIFO status signals are used by the control unit to safely manage readout and initiate subsequent processing.

Once new data is written into the FIFO, the *wr_ack* status flag signals a control unit, which subsequently initiates a read operation, initiating

Table 3
Utilization details.

Resources	Available	Encoder-based TDC	Encoder-based TDC (%)	MLP-based TDC + Systolic Array	MLP-based TDC + Systolic Array (%)
FF	484,800	2535	0.52	5921	1.22
LUT	242,400	1731	0.71	1047	0.43
DSP	1920	1	0.05	215	11.20
BRAM	600	15.50	2.58	196	32.67

Table 4
Power report.

Design Power	Encoder-based TDC	MLP-based TDC + Systolic Array
Dynamic (W)	0.262	0.90
Static (W)	0.480	0.49
Total power (W)	0.742	1.39

the transfer of the data of the data from the FIFO to the I/O Buffer of the accelerator. Directly feeding the SA accelerator from the FIFO output was deliberately avoided for several reasons. First, the TDC output vector must undergo partitioning to enable operation tiling, ensuring compatibility with both the SA dimensions and the expected instruction flow. This necessitates an intermediate pre-processing module before computation. Second, the same input vector must be accessed multiple times since the layer's weight matrix is partitioned to accommodate the SA size. Each partition, once loaded into the SA grid, must be multiplied by the same vector, requiring a persistent memory element to store the vector and facilitate multiple reuse cycles.

Finally, N input vectors are accumulated before computation begins, considering an $N \times N$ systolic array, to fully leverage the SA parallelism and maximize weight reuse. The vector partitioning and memory placement strategy must be carefully designed to maintain mathematical consistency throughout the neural network processing. To facilitate this, the vector partitioning module has been equipped with a counter that activates as soon as the wr_ack flag is asserted (wr_ack = 1). This counter tracks the number of vectors that are partitioned and loaded into SA memory. Once N new vectors are successfully stored, the counter triggers the control unit, which then starts fetching instructions from the instruction memory and initiates the inference process on the TDC output.

6. Experimental analysis

This section presents the experimental evaluation of two TDC architectures. The first approach utilizes the commonly adopted sum1s encoder, while the second is our proposal to integrate the TDC with NNs and the systolic accelerator. Both designs are implemented on an AMD UltraScale 20 nm CMOS KU040 FPGA hosted on the KCU105 development board. In both architectures, the TDC core is built with TDLs using 464 delay elements mapped on CARRY-8 logic resources. The coarse time measurement is performed using a 12-bit counter. For the encoder-based TDC, the entire system operates at 400 MHz. In contrast, the NN-accelerated TDC adopts a multi-clock domain design, where the measurement path runs at 400 MHz, while the processing path operates at 200 MHz, as formerly discussed. The systolic array is configured as a 14 $\times$ 14 grid of PEs, representing the maximum array size supported by tinyTPU. A summary of implementation results, in terms of logic resource utilization for a single-channel TDC instance and estimated power consumption, is provided in Tables 3 and 4. The MLP-based design naturally requires more resources due to the inclusion of the systolic array, which makes use of BRAMs and DSPs for NN inference. Regarding scalability, the architecture is intended to support multi-delay-line or multi-channel configurations by replicating the TDC front-end (TDLs and associated logic) per channel. Since the TDC front-end primarily consumes LUTs/FFs and carry-chain resources, the main practical constraints for scaling are the availability of suitable differential I/O resources and timing closure under the imposed placement/routing constraints required to preserve delay-line uniformity. In contrast, the BRAM/DSP utilization is dominated by systolic array part.

In the target application context, the TDC is intended to operate in a controlled environment, such as a clinical or hospital setting, where power consumption and area utilization are typically secondary concerns compared to latency, throughput, and measurement robustness. In this scenario, trading FPGA resources for increased computational parallelism is an acceptable and often preferred design choice. Accordingly, the proposed approach does not aim at minimizing hardware resource usage, but rather at maximizing inference throughput and enabling real-time, fully on-chip processing. The systolic-array accelerator is therefore employed as a neural-network inference engine, deliberately exploiting a significant portion of the available DSP and BRAM resources to sustain high data rates and low latency. The reported resource utilization should thus be interpreted as a throughput-oriented configuration rather than a minimum-resource baseline. However, this choice does not compromise scalability since the systolic-array architecture can be tailored to different devices and system constraints by

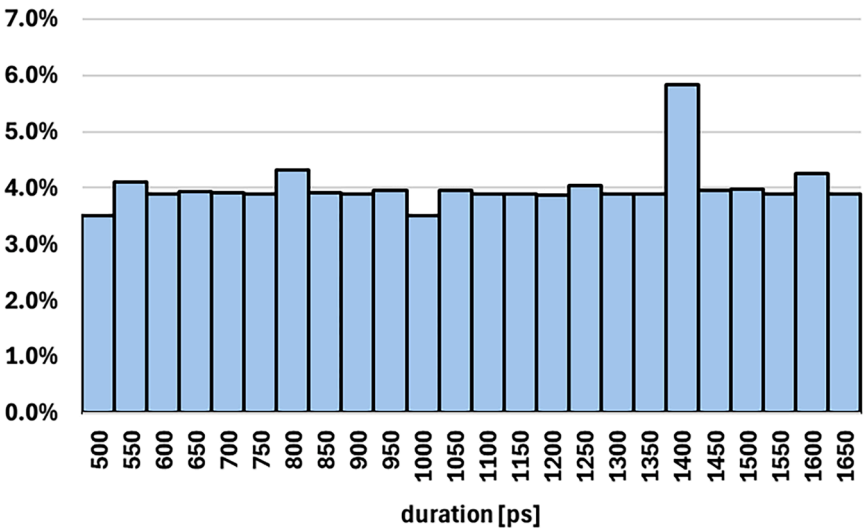

Fig. 7. Class distribution in the dataset.

Table 5
Neural network details.

NN	Hidden Layers	Weights [#]	Inference Time [ms]				MAE[1] [ps]		SAE[2] [ps]	
			GPU	CPU		FPGA				
			Fl32	Fl32	Int8	Int8	Fl32	Int8	Fl32	Int8
MLP	1 FC	60,289	0.09 ±0.06	0.04 ±2.07e-3	0.11 ±0.02	0.03 ± 5e-5	8.65	10.05	8.67	9.12
DNN	5 FC	153,601	0.15 ±7.66e-3	0.77 ±0.16	0.31 ±0.03	0.16 ±1e-4	8.17	9.05	8.61	8.67
CNN	2 Conv + 1 FC	168,737	0.32 ±0.16	0.22 ±1.27e-2	0.45 ±0.05	0.21 ± 1e-4	8.10	9.38	8.53	8.49

[1] Mean absolute error [2] Sum absolute error.

adjusting the array dimensions. Both implementations leverage the custom place-and-route strategies applied to the delay line to minimize variations in the delay introduced by each tap. As introduced previously, achieving symmetrical timing behavior is crucial to reducing errors in the thermometer code. Furthermore, to enable a fair comparison between the two strategies, we want to emphasize that the encoder-based solution and the proposed NN-based solution share the same TDL stage and adopt identical custom placement and routing constraints, so that both architectures operate on identical raw TDL outputs and differ only in the subsequent decoding stages.

A. Calibration

The adopted calibration approach is based on an NN models, trained on a dataset of 344,012 input vectors. These vectors were generated using discrete time intervals with a 50 ps step size, following the timing resolution required by the HONEY project specifications. For each step, about 5000 samples were automatically acquired to account for the intrinsic stochastic behavior of the TDC and to capture a broad statistical representation of its response. The dataset is generated using a dummy design that includes only the delay lines, with same placement of the target ones, and a synchronous counter, all together producing a raw 940-bit binary vector as outputs. The dataset is split into training, validation, and test sets. First, 20 % of the data is reserved as a test set for final evaluation. Of the remaining 80 %, 20 % is allocated to validation and the remaining 80 % is used for training. A two-channel pulse generator was used to emulate precise timing differences between the input signals S1 and S2 used to drive the two delay chains of the TDC. All measurements were collected using a single FPGA board (Xilinx Kintex UltraScale KCU105) under nominal laboratory conditions at room temperature. Data acquisition was repeated on multiple days across independent sessions to check measurement consistency. To verify that the observed behavior is not specific to a single board, we also repeated the same acquisition procedure using the same bitstream on a second KCU105 board of the same type and observed consistent trends; however, the results reported in this work correspond to the primary board used for model training and evaluation (Fig. 7). All results correspond to the same physical FPGA device and the same implemented design/bitstream. No controlled temperature or supply-voltage sweeps were performed; therefore, performance across environmental variations is not experimentally evaluated in this work.

Since the NNs are deployed on an FPGA, they have been trained using QKeras (Coelho et al., 2021), a quantization-aware library that supports the definition of quantized layers. The network uses int8 quantization for weights, biases, and activation functions. Training is performed, using the Adam optimizer with a learning rate of $1 \times e$-5, and the model is optimized to minimize the Mean Squared Error (MSE) between predicted and actual time intervals. MSE is chosen as the loss function, while Mean Absolute Error (MAE) is selected as the performance metric, as the model's objective is to output a continuous timestamp, framing this task as a regression problem. The overall characteristics of the implemented models are provided in Table 5, including the inference time considering hardware execution. Specifically, we run the inference process not only on the FPGA, where we implemented the SA accelerator, but also on the Intel i9-10980XE CPU and Nvidia RTX3060 GPU.

As shown by the execution results reported below, two key aspects emerge regarding FPGA implementation. First, the computations achieve a substantial speed-up attributable to the combination of systolic array topology with the adoption of the weight stationary policy and int8 operations. Specifically, as per WS policy, each processing element of the systolic array retains neuron weights in local registers until all required inputs have been processed, which minimizes memory accesses. Moreover, the column-wise propagation of partial products reduces the overhead associated with register-file or external-memory load/store operations that are typical of CPU and GPU architectures. In addition to that, executing on an FPGA, where on-chip memory is primarily implemented with BRAM of limited capacity, necessitates a reduced network memory footprint. Consequently, the network must be quantized. In this work, quantization aware training is constrained by the target FPGA accelerator, which natively supports only INT8 arithmetic; therefore, the neural network is quantized to 8-bit precision in order to match the deployed hardware implementation. Quantization takes with also the avoidance of floating-point arithmetic, hence further improving the execution time and area costs. It is worth noting, however, that int8 quantization could not be employed on the GPU due to hardware limitations, and therefore, GPU inference is performed as floating-point operations, making a direct comparison less favorable to that platform. Moreover, the speed-up achieved on the FPGA is not only due to the accelerator's int8-oriented design, but also to the fact that the neural network weights are preloaded into the BRAM during bitstream generation. This eliminates any need for model transfer at runtime, and since the weight buffers are physically located close to the computation units, data access incurs significantly lower overhead compared to CPU or GPU implementations.

Second, adoption of quantization, while reducing the latency of the NN, comes with the drawback of accuracy degradation. Specifically, quantization increases the mean error by 16 % and reduces the precision by 5.6 % in the worst case represented by the smallest model.

Another paramount aspect, not captured in the table, is that implementing both the measurement system and the signal-processing logic on a single FPGA device inherently reduces input–output latency. Indeed, in a configuration in which the TDC resides on the FPGA while calibration or result encoding is executed on an external GPU or CPU necessarily incurs data-transfer overhead. This overhead is eliminated by the dedicated design that integrates the two modules within the same chip. In our proposal, the integration effectively enables real-time operation of the platform without any external post-processing steps.

The error values across the different models vary only marginally, whereas model complexity, and consequently execution time and memory footprint, differ substantially. Transitioning from the MLP to either the DNN or CNN more than doubles memory usage and increases execution time by nearly a factor of six. Given the HONEY project's technical specification of a required timing precision on the order of 50

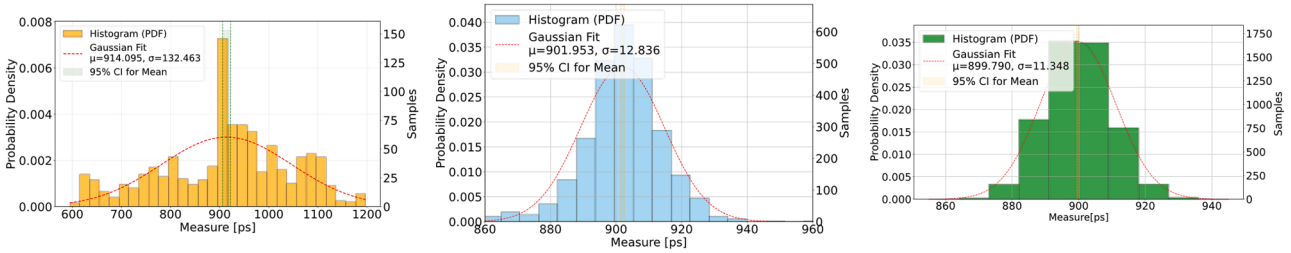

Fig. 8. Example of 900 ps time measurement for the different TDC architectures. In (a) the sums1 encoder-based TDC ($n = 1000$) on FPGA, in (b) the proposed MLP-TDC ($n = 2729$) on FPGA, in (c) the ASIC picoTDC ($n = 5263$).

ps, all three developed models remain well within tolerance, achieving mean errors below 10 ps. Therefore, model selection is primarily driven by design priorities. Because the particle rate is high and events occur in dense beam spills across many sensor channels, we prioritized execution speed to enable time-of-flight measurements as close to real-time as possible, ultimately selecting the simplest and most compact model, the single-hidden-layer MLP.

B. Comparison of NN-based TDC with State-of-the-Art

As previously introduced, the proposed NN-based TDC architecture has been evaluated against a conventional implementation based on a sum1s encoder. Both architectures utilize same input configurations, including the same FPGA I/O pins to receive the S1 and S2 timing signals, and share identical delay line characteristics. To further validate the proposed approach, our TDC is also compared against picoTDC, a high precision 65 nm CMOS ASIC picosecond resolution TDC developed by CERN [21].

The different TDC implementations were evaluated using identical input signals generated by a Pulse Rider PG-1072 to emulate the sensor front-end electronics used in the project (Altruda et al., 2023). Pulse Rider PG-1072 is a dual-output pulse generator capable of producing pulses with rise/fall times below 70 ps and a timing resolution of 10 ps, with an accuracy of $\pm$ (0.1 % + 30 ps). The generator was configured to output 3 V signals at frequencies up to 10 MHz. These signals were then converted to LVDS18 using a translator board (SKYWORKS si53301) (Zhang et al., 2025).

In the MLP-based TDC, the raw 940-bit output vector from the delay line is fed directly into the quantized neural network accelerator, which produces the final time measurement. In contrast, the Sum1s-encoder-based TDC assigns each delay line to a dedicated bit counter, whose outputs are then processed by an encoding stage that maps the number of detected bits to the corresponding time interval in picoseconds.

It is important to emphasize that the goal of this comparison is not to provide a direct performance benchmark between the MLP-based solution and a fully calibrated encoder-based approach. Fine-grained calibration techniques have already been extensively explored in previous works and are known to yield good performance. Instead, the purpose of the following experiments is twofold. First, to demonstrate that the encoder-based approach, without calibration, is inherently sensitive to process variations, leading to significant measurement variability. Second, to highlight that the MLP-based TDC effectively performs an implicit calibration through training, achieving comparable accuracy to ASIC-based solutions, that is used as reference for state-of-the-art solution. Using the same raw 940-bit input vectors as the Sum1s encoder, and without requiring any pre-processing or manual calibration, the neural network significantly reduces the standard deviation of the measured time intervals, achieving measurement precision comparable to the picoTDC ASIC and confirming the effectiveness of the proposed approach.

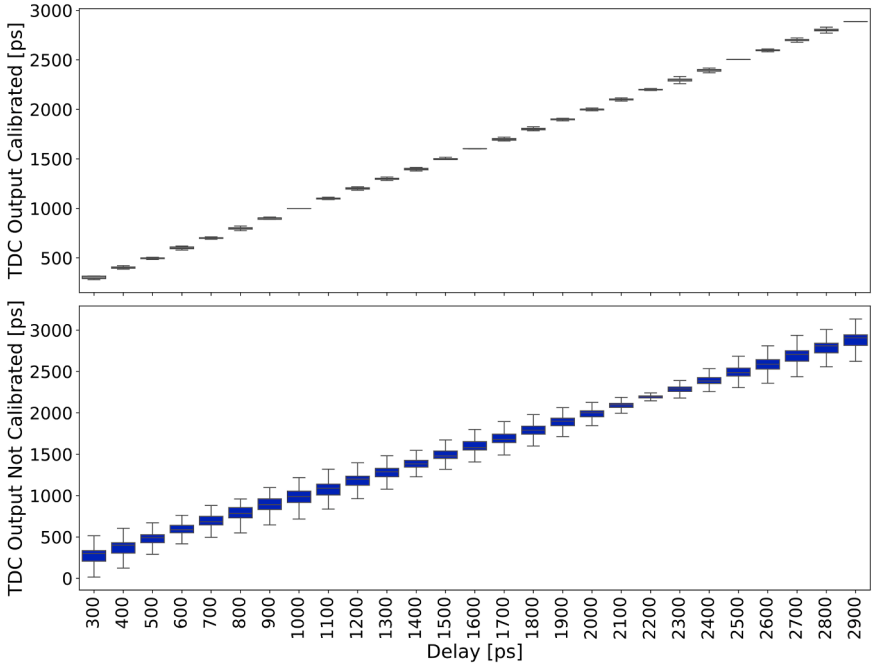

Fig. 9. Error distribution over the different measurement for the different TDC architectures. In (a) the sums1 encoder-based TDC on FPGA, in (b) the proposed MLP-TDC on FPGA, in (c) the ASIC picoTDC.

Table 6
Comparative summary of the proposed FPGA-based TDC.

Method	Resolution (ps)	Experimental σ (ps)	Latency	Training/calibration time	Hardware Device
sum1s encoder (our baseline)	6.03	133.2	100 ns (pipelined)	High	FPGA
MLP-based TDC + Systolic Array	6.03	13.5	30μs	Low	FPGA
picoTDC (1)	3 (fine mode)	3.7(1.35)	N/A	High	ASIC (65 nm CMOS)

As an example, in Fig. 8, we report three histograms showing measurements of a 900 ps interval using the different TDC architectures: (a) the FPGA TDC with a traditional *sum1s* encoder, (b) the FPGA TDC using our proposed neural network MLP encoder, and (c) the ASIC picoTDC. The encoder-based FPGA TDC in (a) shows poor precision ($\sigma \approx 134.9$ ps) and significant skew, reflecting the typical issues of FPGA-based delay lines due to nonlinearity, mismatch, and susceptibility to process variations. Differently, the NN-based TDC in (b) achieves a precision improvement of over 10.5 ×, reducing the standard deviation to ≈12.8 ps. Additionally, the mean value aligns with the expected 900 ps ($\mu \approx$ 902 ps), reached by the majority of the measurements. This demonstrates the neural network's ability to effectively learn and compensate for the non-uniformities in the FPGA delay line. Remarkably, the performance of the NN-based TDC is now comparable to that of the ASIC TDC in (c), which achieves $\sigma \approx 11.3$ ps.

The second set of histograms in Fig. 10, illustrates the error distribution across a wide range of time interval measurements, from 50 ps to 1500 ps, for the same three TDC architectures. The neural network-enhanced TDC delivers a significantly narrower and well-centered error distribution, with a standard deviation of only 13.6 ps and around 2 ps mean error when considering measure over the wide dynamic range. These results indicate that in general the neural network-enhanced TDC closely approximates the performance of the ASIC-based TDC, both in terms of accuracy and stability (Fig. 9). This suggests that the proposed approach not only reduces measurement variability but also effectively compensates for systematic deviations, making it a strong candidate for high-precision timing applications.

To quantitatively assess the significance of this improvement, we performed a statistical comparison between the encoder-based and NN-based error distributions. Normality was tested using the D'Agostino–Pearson normality test. The results indicate that the error distributions deviate from Gaussianity, justifying the use of non-parametric statistical tests for comparative analysis. Since the distributions deviate from normality, statistical significance was evaluated using the non-parametric Mann–Whitney U test, which confirms that the reduction in error achieved by the NN-based approach is statistically significant ($p < 10^{-16}$). In addition, the practical relevance of this improvement was quantified using the non-parametric effect size Cliff's delta, which yields $\delta = 0.86$, indicating a very large effect.

Compared with the approach reported in (Garzetti et al., 2024), our method shows a slightly higher standard deviation in the timing measurement. However, the neural network we deploy is far more compact, only 65 neurons in total, versus their ten-layer architecture with thousands of neurons. If an application demands higher precision, our framework can seamlessly switch to any of the alternative neural-network configurations listed in Table 5 (or even a new custom model). Indeed, updating the accelerator buffer with the new weights and issuing the corresponding low-level instructions is sufficient to load and run a different model thanks to the decoupling and flexibility provided by the use of the systolic array. This flexibility comes while achieving a 96.8 % reduction in network size relative to (Garzetti et al., 2024), yet maintaining comparable timing accuracy.

Table 7
Experimental setting.

Experimental Equipment and Settings	Characterization / Setting Used In This Work
FPGA Board	AMD Kintex UltraScale FPGA KCU105 Evaluation Kit
Pulse Rider PG-1072 (dual-output) Pulse Generator	timing resolution 10 ps - rise/fall < 70 ps - accuracy ± (0.1 % + 30 ps) - output up to 10 MHz
Signal characteristics	3 V generator outputs - frequency 1 to 10 MHz (different implementation) - Channel 2 is the inverted (complementary) of Channel 1
Skyworks Si53301	Configured to generate LVDS18 (between pulse generator and FPGA)
Cables	Two pair of matched SMA coaxial cables, 0.5 m each
Hightech Global SMA/LVDS FMC Module	Use for connecting the signals to FMC Ios of the FPGA board

Our model also differs fundamentally in data handling. In Garzetti et al. (2024)), the MLP processes only the delay-line output, so an extra post-processing step is required to merge the inferred fine time with the coarse counter value to obtain the final timestamp. By contrast, our network processed both the delay-line output and the counter value simultaneously, learning the complete time-reconstruction task and producing the final timestamp directly eliminating the need for external combination logic. Finally, it is crucial to remark that inference in Garzetti et al. (2024)) is performed on a host PC, whereas our design integrates the model into a dedicated hardware platform surrounding the TDC on the same chip, enabling real-time on-chip inference without external computation or dependence. In Table 6, we summarize the performance improvement of the proposed approach over the encoder-based baseline and include picoTDC as an ASIC reference. The picoTDC precision values reported in Table 6 are taken from Altruda et al. (2023). For completeness, we note that in our own measurements (Fig. 10) picoTDC exhibited a standard deviation of 11.9 ps, which also include the uncertainty and limitations of the laboratory equipment. In

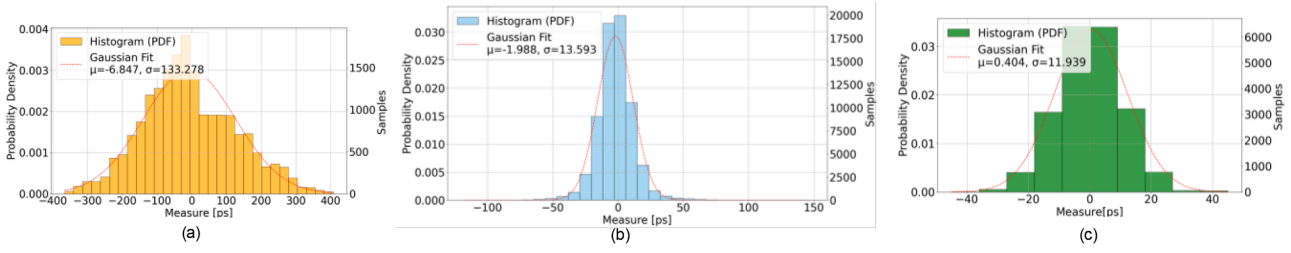

Fig. 10. Error distribution between non calibrated TDC vs calibrated TDC.

addition, Table 7 consolidates the experimental setup and instrumentation used in this work.

7. Conclusion

This work presented an FPGA-based TDC achieving picosecond-level precision by integrating a hardware-accelerated neural network for on-chip timestamp reconstruction. By replacing conventional decoding and explicit calibration with a local inference engine, the proposed design delivers accuracy comparable to ASIC-based solutions while preserving the flexibility of programmable logic and eliminating host-side post-processing. The learning-based approach is well suited to compensate for device- and implementation-related non-linearities, such as those introduced by routing variations and process-dependent effects. Environmental variations, including temperature and supply voltage changes, manifest as systematic distortions in the raw delay-line outputs and can be accommodated through model retraining and redeployment without requiring hardware modifications. While the trained model is device-specific, due to chip-to-chip variability in FPGA routing and delay characteristics, retraining can be performed offline and updated parameters can be deployed on-chip with minimal overhead. A systematic experimental characterization of long-term drift, temperature and voltage sensitivity, and retraining frequency across multiple FPGAs and varying environmental condition represents an important direction for future work.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This research is part of HONEY (Hybrid ONline technology for particle therapy), a PRIN (Progetti di Ricerca di Interesse Nazionale) project funded by the Italian Ministry of University and Research (MIUR), CUP number I53D23001280006. The project is conducted in collaboration with Politecnico di Torino, INFN, and the University of Torino, with valuable support from CNAO (National Center for Oncological Hadron Therapy).

Data availability

Data will be made available on request.

References

S. Altruda et al."PicoTDC: A flexible 64 channel TDC with picosecond resolution" 2023, *JINST* 18 P07012, 10.1088/1748-0221/18/07/P07012.

AMD, FIFO Generator v13.2 Product Guide (PG057), Document ID: PG057, Version 13.2, Oct. 4, 2017. [Online]. Available: Docs.amd.com/v/u/en-US/pg057-fifo-generator.

Bayer, E., & Traxler, M. (2011). A high-resolution (< 10 ps RMS) 48-channel time-to-digital converter (TDC) implemented in a field programmable gate array (FPGA). *IEEE Transactions on Nuclear Science, 58*(4), 1547–1552. https://doi.org/10.1109/TNS.2011.2141684

Castelvero, L., López Grande, I. H., & Pruneri, V. (2024). High-performance time-to-digital conversion on a 16-nm ultrascale+ FPGA. *IEEE Access : Practical Innovations, Open Solutions, 12*, 149569–149579. https://doi.org/10.1109/ACCESS.2024.3477295

Coelho, C. N., Kuusela, A., Li, S., et al. (2021). Automatic heterogeneous quantization of deep neural networks for low-latency inference on the edge for particle detectors. *Nature Machine Intelligence, 3*, 675–686. https://doi.org/10.1038/s42256-021-00356-5

Fan, G., Sabri, A. Q. M., Rahman, S. S. A., et al. (2025). Emerging trends in graph neural networks for traffic flow prediction: A survey. *Archives of Computational Methods in Engineering, 32*, 4811–4855. https://doi.org/10.1007/s11831-025-10286-9

Fishburn, M., Menninga, L. H., Favi, C., & Charbon, E. (2013). A 19.6 ps, FPGA-based TDC with multiple channels for open source applications. *IEEE Transactions on Nuclear Science, 60*(3), 2203–2208. https://doi.org/10.1109/TNS.2013.2241789

[8]J. Fuhrmann, "Implementierung einer tensor processing unit mit dem fokus auf Embedded Systems und das Internet of Things", Germany, 2018., http://hdl.handle.net/20.500.12738/8527.

[9]F. Garzetti et al., "Plug-and-play TOF-PET module readout based on TDC-on-FPGA and gigabit optical Fiber network," *2019 IEEE nuclear science symposium and medical imaging conference (NSS/MIC)*, Manchester, UK, 2019, pp. 1–4, 10.1109/NSS/MIC42101.2019.9059966.

Garzetti, F., et al. (2024). Novel machine learning-driven optimizing decoding solutions for FPGA-based time-to-digital converters. *Measurement, 238*, Article 115313. https://doi.org/10.1016/j.measurement.2024.115313. ISSN 0263-2241.

Genschow, D. (2015). A time to digital converter for use in ultra wide band radar sensor nodes. In *2015 IEEE topical conference on wireless sensors and sensor networks (WiSNet)* (pp. 38–40). https://doi.org/10.1109/WISNET.2015.7127419

Hua, Y., & Chitnis, D. (2022). A highly linear and flexible FPGA-based time-to-digital converter. *IEEE Transactions on Industrial Electronics, 69*(12), 13744–13753. https://doi.org/10.1109/TIE.2021.3128912

Jain, R., Dhingra, S., Joshi, K., Rana, A. K., & Goyal, N. (2023). Enhance traffic flow prediction with real-time vehicle data integration. *Journal of Autonomous Intelligence, 6*(2), 574. https://doi.org/10.32629/jai.v6i2.574

Khullar, V., Singh, H. P., Miro, Y., Anand, D., Mohamed, H. G., Gupta, D., Kumar, N., & Goyal, N. (2022). IoT fog-enabled multi-node centralized ecosystem for real time screening and monitoring of health information. *Applied Science, 12*, 9845. https://doi.org/10.3390/app12199845

Kim, J., Jung, J. H., Choi, Y., Jung, J., & Lee, S. (2023). Linearity improvement of UltraScale+ FPGA-based time-to-digital converter. *Nuclear Engineering and Technology, 55*(2), 484–492. https://doi.org/10.1016/j.net.2022.10.010. ISSN 1738-5733.

Liu, C., et al. (2016). 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)* (pp. 1–3).

Liu, C., & Wang, Y. (2015). A 128-channel, 710 M samples/second, and less than 10 ps RMS resolution time-to-digital converter implemented in a Kintex-7 FPGA. *IEEE Transactions on Nuclear Science, 62*(3), 773–783. https://doi.org/10.1109/TNS.2015.2421319

Lusardi, N., et al. (2022). High-performance time-to-digital converter IP-core for Xilinx Ultrascale/Ultrascale+ FPGAs. In *2022 IEEE nuclear science symposium and medical imaging conference (NSS/MIC)* (pp. 1–5). https://doi.org/10.1109/NSS/MIC44845.2022.10399311

Ranjbar, S., et al. (2026). Performance Analysis of In-Beam PET Range Verification System for Carbon Ion Beams. *in IEEE Transactions on Radiation and Plasma Medical Sciences, 10*(1), 137–143. https://doi.org/10.1109/TRPMS.2025.3571308

Roy, N., Nolet, F., Dubois, F., Mercier, M.-O., Fontaine, R., & Pratte, J.-F. (2017). Low power and small area, 6.9 ps RMS time-to-digital converter for 3-D digital SiPM. *IEEE Transactions on Radiation and Plasma Medical Sciences, 1*(6), 486–494. https://doi.org/10.1109/TRPMS.2017.2757444

Singh, T. P., Gupta, S., Garg, M., Gupta, D., Alharbi, A., Alyami, H., Anand, D., Ortega-Mansilla, A., & Goyal, N. (2022). Visualization of customized convolutional neural network for natural language recognition. *Sensors, 22*, 2881. https://doi.org/10.3390/s22082881

Skyworks Solutions, Inc.. (2021). Ultra-low additive jitter fanout clock buffers with up to 10 universal outputs from any-format input and wide frequency range from 1 MHz to 725 MHz", Si5330x data sheet, rev 1.0. Online] https://www.skyworksinc.com/-/media/Skyworks/SL/documents/public/data-sheets/si5330x-datasheet.pdf accessed April 20th, 2025.

Umuroglu, Y., Fraser, N. J., Gambardella, G., Blott, M., Leong, P., Magnus, J., & Kees, V. (2017). FINN: A framework for fast, scalable binarized neural network inference. In *Proceedings of the 2017 ACM/SIGDA international symposium on field-programmable gate arrays (FPGA '17)* (pp. 65–74). Association for Computing Machinery. https://doi.org/10.1145/3020078.3021744.

Villa, F., et al. (2013). CMOS single photon sensor with in-pixel TDC for time-of-flight applications. In *2013 IEEE Nordic-Mediterranean workshop on time-to-digital converters (NoMe TDC)* (pp. 1–6). https://doi.org/10.1109/NoMeTDC.2013.6658230

Xilinx, A. M. D. (2023). KCU105 evaluation board for the Kintex UltraScale FPGA: User guide, UG917 (v1.7). May https://www.xilinx.com/support/documents/boards_and_kits/kcu105/ug917-kcu105-eval-bd.pd.

Zhang, H.-F., et al. (2012). A real-time QKD system based on FPGA. *Journal of Lightwave Technology, 30*(20), 3226–3234. https://doi.org/10.1109/JLT.2012.2217394

Zhang, Z., Zhang, M., Zhang, Y., He, Z., & Wan, M. (2025). A digital compatible offset canceled latch-styled sense amplifier used for PUF sensing. *in IEEE Transactions on Circuits and Systems I: Regular Papers*. https://doi.org/10.1109/TCSI.2025.3603286