IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 10, 2022, accepted April 11, 2022, date of publication April 22, 2022, date of current version May 6, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3169760

High-Performance Computing of Real-Time and Multichannel Histograms: A Full FPGA Approach

ANDREA COSTA, (Member, IEEE), NICOLA CORNA, (Member, IEEE), FABIO GARZETTI, (Member, IEEE), NICOLA LUSARDI, (Member, IEEE), ENRICO RONCONI, (Member, IEEE), AND ANGELO GERACI, (Senior Member, IEEE)
Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB), Politecnico di Milano, 20133 Milan, Italy

Corresponding author: Andrea Costa (andrea1.costa@polimi.it)

ABSTRACT In a world heading towards applications, in science and industry, based on big data processing, the ability to elaborate streams of data is becoming more important every day. Applications of various natures, ranging from biology to chemistry, from medical imaging to spectroscopy, need systems able to detect, process and store huge amounts of data in real-time. In this context, techniques such as histogramming come into play. Histograms are able to represent data shapes and retrieve statistical information, favoring further processing. This kind of processing is usually done with the help of general purpose processors, relying on temporal computing, with their pros (simplicity and fast operating frequencies) and cons (inability to exploit parallel computation). Both Industry and Academia have, however, proposed many solutions to this need, delivering histogram generators both in full-custom Application-Specific Integrated Circuits (ASICs) and Field-Programmable Gate Array (FPGA) IP-Cores, preferred over processors thanks to their superior parallel computing power. In this work, we present a full FPGA approach to this issue, resulting in a high-performance IP-Core for generating and managing multiple histograms in real time, each supporting a data flow up to 224 MSps, with a strong focus on overall flexibility and area efficiency, using as low as < 250 LUTs and 300 FFs resources for a 16 bit-wide, 4096-bin histogram implementation. The IP-Core has been successfully integrated and validated in a high-performance time-mode application: an FPGA-based Time-to-Digital Converter. The resulting IP-Core is, more in general, a single solution perfectly adaptable to any field of application and FPGA device.

INDEX TERMS Field-programmable gate array (FPGA), histograms, real-time systems, IP-Cores, throughput.

I. INTRODUCTION

Compressing data for managing it in huge amounts can be often accomplished [1] most conveniently by histograms [2]. As is well known, for example, histograms keep track of the frequency of events [3] by storing the information in a user-defined memory which, in first approximation, can be dimensioned independently of the number of events. Nowadays, the need to manage big data, that can even be spread over extremely extended value dynamics, is more and more frequent [4]. This is the case for many applications that can provide big advantages from the possibility of obtaining a more compact representation of the observed phenomenon. Moreover, a large number of applications have abandoned deterministic models in favor of stochastic ones [5], [6], based on predictive algorithms, such as weather forecasting [7] and traffic prediction [8], just to name a few. The information needed can be extracted and used in several ways, e.g. by computing statistical moments, from mean-value to variance and higher moments [9]. In fact, from a stochastic point of view, the normalized version of a histogram is the best approximation of the random process p that generated the sequence [10]. In this scenario, the role of data analysis and processing using histograms assumes an increasingly fundamental role, since they allow a straightforward extraction of the statistics of the p process that generated the stream of data considered.

Moreover, due to their simplicity, histograms find application in an endless range of metrological fields that span from Industry to Research. Just think about computer vision, where the histogram of the gray-scale image and its variance are used for real-time image recognition [11], [12].

The associate editor coordinating the review of this manuscript and approving it for publication was Jiafeng Xie.

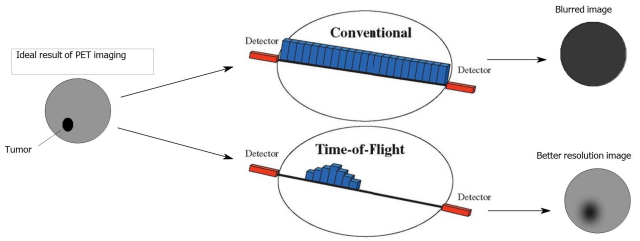


FIGURE 1. Comparison between PET and TOF-PET, given the ideal image (on the left). Undoubtedly the TOF-PET provides a better resolution in the image, thanks to the possibility to restrict the position of annihilation to just a part of the Line of Response (LOR). Both procedures are based on histogram use.

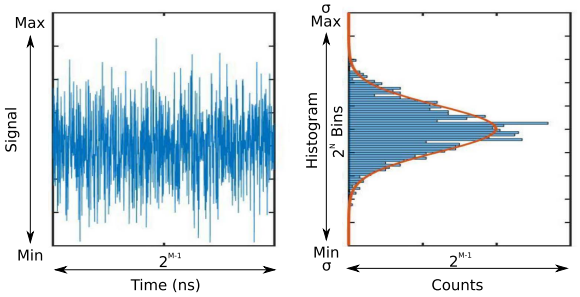


FIGURE 2. On the left: temporal evolution of the signal of interest. On the right: histogram of the signal.

Considering the scientific research environment, histograms are widely used in time-based experiments, being at the basis, for instance, of nuclear physics investigations. In this scenario the histogram tool performs energy spectra, measures gamma-photons arrival time distribution in Time-of-Flight Positron Emission Tomography (TOF-PET) [13], Time-Resolved Spectroscopy [14], Time-Correlated Single Photon Counting (TCSPC) [15], Time-of-Flight (TOF) Rangefinder such as Laser Rangefinders [16] (LR), Time-of-Flight Mass Spectrometry [1] (TOF-MS), and so on. Figure 1 is just an example of a histogram application for the acquired timestamp in TOF-PET analysis.

As is well known, in the digital world each piece of quantized data is represented by means of a N-bit wide word, that can be mapped to the analog world by simply multiplying it by the so-called Least Significant Bit (*LSB*). In these terms, the Full-Scale Range (*FSR*) of the analog values that can be quantized extends to $2^N \cdot LSB$.

In this scenario, if we built a histogram, the *FSR* of the data sequence should be divided into *LSB* wide classes, a.k.a. bins. On the other hand, a digital memory composed of 2^N cells can properly store a histogram characterized by $\frac{FSR}{LSB} = 2^N$ bins.

As regards the digital memory that makes up the histogram, each cell is characterized by data width M, which defines the maximum number of counts ($Count_{max} = 2^M - 1$) that a single bin can store without saturation. The M parameter defines the accuracy of the histogram, in the sense that the greater the number of samples in the histogram, the more accurate is the estimation of the statistical process p. A representation of the histogramming process of a generic signal can be seen in Figure 2.

The most intuitive way to build up a histogram is based on temporal computing processors, e.g. Central Processing Units (CPUs) or Graphics Processing Unit (GPUs), where the main advantage is the simplicity of the algorithm but, as a drawback, we have a limit on the maximum number of threads (e.g., histograms) run in parallel. However, modern applications exploit multi-channel solutions, where huge (from tens up to hundreds) processing cores, in this case histogramming engines, run independently. In this scenarios spatial computing architectures are mandatory. An initial idea could be the realization of an Application-Specific Integrated Circuit (ASIC), with the purpose of producing the histogram of an input stream of data as output [17], but this approach does not fit with the lower time-to-market request for the application: in fact, academic and also industrial R&D departments look for systems based on programmable logic for fast-prototyping. In this sense, the aim of this work is the realization of a ready-to-use, FPGA-based IP-Core architecture for real-time, multi-channel histogramming with latency as low as 2 clock cycles and a high data rate, up to 224 MSps guaranteed for any configuration, usable for and adaptable to various modern applications. The development of the IP-Core is based on VHSIC Hardware Description Language (VHDL), a hardware description language that can model behavior and structure of digital systems independently of the physical device they are put on, allowing many different technologies to host the IP.

The paper is organized as follows; in Section II the trend of moving to parallel computing solutions and the state-of-the-art in the field of histogram computation at high-performance are introduced; in Section III, the technicalities and characteristics of the proposed architecture are discussed. Finally, in Section IV the experimental validation in time mode experiments of the IP-Core is presented.

II. TREND OF IMPLEMENTATION STRATEGY AND STATE-OF-THE-ART

The easiest way to build histograms is by making use of temporal computing, by means of general–purpose processors like CPUs [18] and GPUs [19]. However, we are seeing an increasing spread of multi–channel applications where parallel computing and multi–thread processing are a must in different branches of research, from quantum experiments [20] to nuclear physics [21], from machine learning [22] to astronomy [23]. Limiting ourselves, as an example, to the field of metrology, we are witnessing a continuous increase in the number of parallel input channels in instruments such as oscilloscopes, function generators and, more generally, acquisition set-ups [24]. A temporal computing approach, based on sequential processing, becomes intrinsically inefficient in these kinds of applications. This issue has led research to look towards a lower level approach, where interfacing with the source of data can be exploited at full speed and

processing solutions can be built for the specific application, supporting multiple inputs, each one provided with a dedicated processing core [25], [26]. This non-trivial, but far higher performing way of dealing with these needs relies on the use of dedicated architectures that can be implemented on FPGA devices.

The use of devices featuring spatial computing, i.e. parallel architectures, makes dramatically increasing the processing capabilities possible, lowering the latency overhead, because of the radically different approach to the problem. They exploit parallel computation, and their design is completely customizable, in order to perfectly adapt to specific needs. In addition to FPGAs, a new kind of System-on-Chip (SoC) is quickly becoming the basis of high-performance applications, merging the benefits of Programmable Logic with the flexibility and the large open source software base of a Processing System [27]–[29].

Even taking into account some end consumer applications, like image and video processing for Computer Vision [30] and Automotive [31] uses, they greatly rely on histogramming techniques. The common denominators of all histogram applications are, therefore, the large amount of data and, more importantly, data rates: it is essential to be able to cope with these rates with the real-time processing, to avoid generating of bottlenecks and to guarantee the required performance, that is ever increasing [32].

As you can imagine, a conventional temporal-computing CPU approach, thanks to the general speedup of digital devices, offers both good performance and widespread knowledge of the programming methods, but lacks the possibility of effectively extending the processing to multichannel application via parallelization, maintaining the same performance in terms of throughput. In this case, the most interesting approaches, that make this feature one of their main strengths, are GPUs and FPGAs. In fact, they adapt perfectly to parallel computation needs, offering higher performance when compared to CPUs. Their drawback is, essentially, the more complicated programming techniques. For this reason, in the case of GPUs, tools like CUDA [39] have been developed, to allow a similar programming style to CPUs, for accelerated computing. However, programming becomes non trivial in the case of FPGAs, due to the expertise in digital electronics circuits required and the in-depth knowledge of the specific hardware support. In order to overcome these limitations, nowadays, FPGAs can also be interacted with via newly developed High Level Synthesis (HLS) tools, that make the programming style more CPU-like; but the best performance is, however, still achieved only with a classical description of the hardware that the design is targeting, by using Hardware Description Language (HDL) tools, e.g. VHDL and Verilog.

It can be noted how, in [34], for example, the implemented method is based on an external GPU setup. This guarantees (based on the GPU performance) large parallel processing capabilities, in terms of overall throughput, but also implies that a structure that is able to readout, communicate and send the data to a host PC at those high data rates (10Gb/s) needs to be put in place: a high speed networking card and cables that interface with the data-generating FPGA, a costly and power hungry dedicated GPU to process the data, all running on a host computer that needs to be powerful enough to keep up with the data rates involved. All of this increases the overall system complexity, power, cost and flexibility. Moreover, a limiting factor in the development of general purpose histogramming algorithms could be the bus width of the GPU architecture of choice.

In this sense, the most convenient way of performing this kind of processing lies in the FPGA domain [40]: in fact, the possibility of directly building a histogramming IP-Core into the hardware and replicate it to exploit maximum parallelism, as needed by the application, provides the chance to keep the system simpler and the processing to be done closer to the data generation, in real-time. The parallelization limit is generally related to the device in use, in terms of the number of resources available.

Moreover, in an environment which probably features some detector generating signals that need to be acquired and processed, it is of fundamental help to be able to embrace at least initial processing in real-time, just after the acquisition chain [41]. In this context, FPGAs make it possible to build ''cores'' for data processing side by side with Data AcQuisition (DAQ), giving rise to a powerful combination that, in many cases, ensures no bottlenecks due to data transfer off chip, a phase that is mandatory when leaving the processing to an external unit such as a GPU. Additionally, in some circumstances, a real-time, low latency bidirectional communication may be needed between the DAQ and the data processing units, in order to enable the system to be used in a feedback configuration, using the processed data to adjust parameters during acquisition [42]. This can be easily achieved if, once again, the DAQ section, which is generally developed on FPGAs, is assisted by a dedicated processing unit on the very same chip.

A brief comparison of different approaches that imply computation of histograms is shown in Table 1. It can be noted how, as previously introduced, histograms are involved in different fields: the top part of the table refers to solutions for time domain experiments ([32]–[35]), while the lower part features applications in the field of computer vision and image processing ([11], [12], [36]–[38]). The implementation trend results in FPGAs as the dominating technology and strictly follows the field of application, making the listed solutions not very suitable for an easy cross-domain utilization, but rather for operating in the environment they were created for in the first place.

For the reasons described, in this work an FPGA-based architecture for the construction of histograms is presented, aiming for re-programmability, multi-channel capabilities (i.e. low resource utilization), low latency, high throughput and maximum flexibility in terms of parameters setting, adaptability among various measurement fields and different FPGA technologies.

TABLE 1. Comparison of applications and related data rates.

Reference	Application	Data rate	Processing Unit
[32]	TOF rangefinder	1.6 Gb/s ($50MHz \times 32bit$)	FPGA
[33]	TOF rangefinder	1.92 Gb/s ($240MHz \times 8bit$)	FPGA
[34]	TCSPC	10 Gb/s	GPU
[35]	TOF-PET	2.5 Gb/s	FPGA
[36]	Computer vision	800 Mb/s	FPGA
[37]	Image processing	2.24 Gb/s	FPGA
[11]	Image recognition	3.2 Gb/s	FPGA
[38]	Image processing	400 Mb/s	FPGA
[12]	Image processing	2.96 Gb/s	FPGA

Various FPGA-based solutions for computation of histograms are present in modern scientific literature. A significant boost was given by real-time computer vision applications; in fact, the most effective real-time image recognition algorithms are based on the computation of histograms of the acquired pictures. The explosion in real-time histogram computation on parallel computing happened around the beginning of the new millennium [43], as the technological nodes made effective hardware primitives (e.g., BRAM, DSP) available to a Programmable Logic Device (PLD), i.e., FPGA and SoC. Moreover, the FPGA approach is also used as an intermediate step to validate the hardware design, with the final scope of implementing it as an ASIC [44].

In this sense, referring to FPGA-based solutions in scientific literature, it is possible to extract the main Figures-of-Merits (FoMs) of a histogram. Referring to Section I, the first that can be derived are the maximum number of bins 2^N, a.k.a. number of values on the abscissa, and the maximum number of counts that each bin can have $2^M - 1$, a.k.a. values on the ordinate. In this context, as the reader can immediately understand, the availability of a memory, e.g. BRAM in Xilinx FPGAs, with a minimum storage capacity of $(M - bit/word) \times (2^N - word)$ is mandatory. Moreover, a proper increment mechanism of the selected bin, e.g. adder or DSP, has to be present. The pipeline introduced by the memory and the increment mechanism determine the latency (L, measured in clock cycles), the maximum rate (R, measured in MSps), and the system clock, of the system (F_{CLK}, measured in MHz), which are additional fundamental FoMs Last but not the least, the total area occupancy is yet another important parameter to consider.

As the reader can easily understand, a larger M imposes the use of wider increment mechanisms, that are characterized by slower propagation delays. Following the same concept, larger N forces to work with larger memories and address mechanisms, which are also characterized by slower propagation delays. In this scenario, a pipeline approach is mandatory in order to speed up the system, at the cost of a larger area occupancy. A possible solution to lessening this trend could be reducing of the maximum input rate in relation to the clock frequency, as done in [38]. On the contrary, the solution presented in [12] uses flip-flops, instead of classical BRAM, in order to memorize histograms, making it possible to work at high-frequency without the need of a pipeline, thus saving area.

Table 2 summarizes the most effective results in FPGA-based histogram modules, focusing on the FoMs presented. In [36], the pipeline composed of the memory and the increment mechanism has a duration of 4 clock cycles, with 2 additional cycles for the read-out, reaching a total pipeline of 6 clock cycles. Such a long pipeline makes it possible to split the combination block, in order to guarantee a low propagation delay, making it possible to work with a clock up to 100 MHz,with $N = 8$ and $M = 32$.

A different approach leading to a fast histogram computation is the parallel histogram array (PHA), which avoids to rely on a memory to accumulate histograms, rather using an array of registers, like in [37]. Here we have $N = 8$ and $M = 22$; the pipeline, here, is larger, allowing use of a 280 MHz clock but using 22.1% of the available resources. This structure allows, in fact, for a large possibility to parallelize the structure, independently of the number of access ports to a memory, at the expenses of a much larger resource overhead as the histograms dimensions grow, with respect to a memory-based architecture. This last drawback keeps us from choosing the histogram array architecture, as we are aiming for a general purpose, area saving and efficient solution that can be implemented alongside whatever other logic a user needs.

Another similar, so called Parallel Array Histogram Architecture (PAHA) is described in [45], and shows the same trend of implementation. In [11] a relaxed pipeline is used: in this way 4 histograms with $N = 8$ clocked at 100 MHz are implemented with an occupation area of the 18%. In [38], a low area consuming solution is opted for, which is able to work up to a 200 MHz clock; this is paid for by a dead-time of 4 clock cycles to update the histogram, reducing the processing rate to 1/4 of the clock frequency. Memory-based implementations can leverage, instead, low area occupation, with the risk of sacrificing the maximum synthesizable frequency, like in [46] and [47].

All of these different histogram implementations, are in fact developed in order to fulfill the needs of specific applications, whether it is high throughput, large number of bins and reduced counts per bin, or vice versa, latency-critical real time environment or more relaxed setups, failing to deliver

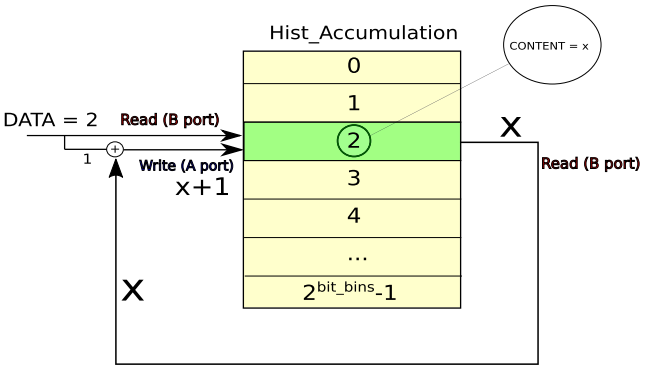


FIGURE 3. Schematic view of the process updating the third bin, "2", of the histogram that contains a value equal to the previous occurrences belonging to current input DATA, given the assumption of making the data value coincide with the address of the corresponding bin of the histogram. The number x of data contained in the bin is incremented by 1.

a real general purpose and flexible solution with the needed performance.

III. IMPLEMENTATION CHALLENGES

The aim of this work is the complete design of a histogramming, memory-based architecture in a Xilinx FPGA device, organized as a tunable IP-Core in term of N and M, aiming for maximum performance in terms of latency, clock frequency, rate and low area occupation. When it comes to building a hardware that computes histograms, the whole process can be thought of simply as the accumulation on the bins of occurrences of the data values, but digging deeper into the design some issues arise that need to be faced properly.

A. BASICS OF HARDWARE HISTOGRAMMING

As mentioned, an accumulation mechanism is needed in order to create a histogram out of single pieces of incoming data. In principle, as one piece of data arrives, the accumulation can be performed by simply increasing the content of the bin corresponding to the data value by one unit.

This mechanism is as simple as it is delicate to implement while maintaining high process efficiency. In fact, physical delays associated with the flow of data in the architecture require careful design [48].

The implemented accumulation process is depicted in Figure 3.

The process for updating the histogram each time input data arrives is made up of three steps:

1) Get valid data and consider its value as the address of the related histogram bin.
2) Read the value of the data already observed corresponding to the selected bin.
3) Increase the number of observed pieces of data data for the selected bin by one.

[5]LUTs and FFs values are, if not explicitly specified, derived and adapted to Xilinx 7 series equivalent (i.e. "Logic Elements" in [36] are traduced to Slices from Xilinx, each containing 4 LUTs and 8 FFs)

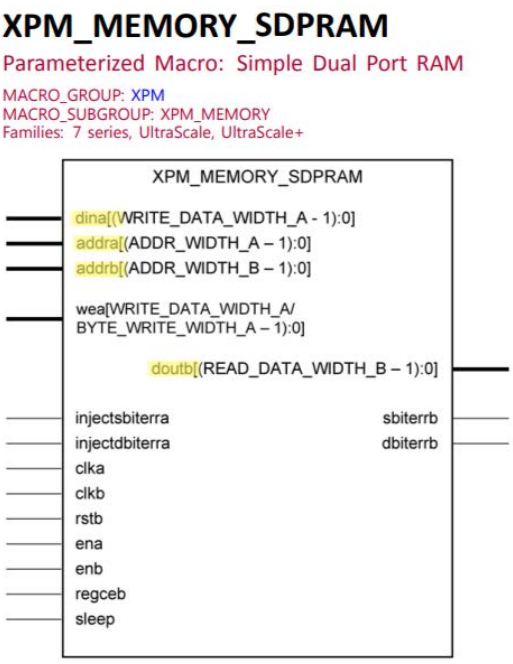


FIGURE 4. SDPRAM resource available in Xilinx FPGAs. As the I/O pin descriptions show, resources labeled as A (on the left) are dedicated to writing and resources labeled as B (on the right) to reading.

This process can be implemented by using an adder in synergy with a Block RAM (BRAM) of the FPGA device, configured in Simple Dual Port operating mode [49]. In this way, read and write ports are available and can be used by different processes at the same time, in order to correctly read the incoming data and then update the content of the memory. Xilinx provides a macro for this purpose, called XPM_MEMORY_SDPRAM (Simple Dual Port RAM, Figure 4), in which Port-A is used to address the write operations, and Port-B manages the read operations.

The second and third steps involve, first, a read and, then, a write operation. As mentioned, the implementation phase makes it not possible to realize the update process of the bin based on the simply described theory. In fact, a latency of at least 1 clock cycle is needed for the memory to be read. This leads to the need of properly buffering the incoming data, as it is also used as the address for the write operation, asserted the clock cycle that follows the read one.

Another big problem that needs to be addressed, when talking about high-performance applications, is timing closure [50], [51]: a design that is required to run at high-frequencies, guaranteeing large bandwidths, will face the constraint of having a limited, short amount of time between one sensitive clock edge and the following one, in which it must complete the required processing that leads from one synchronous element (Flip-Flops, RAMs, Latches, and so on) to the following one. In order to achieve timing closure the usage of thorough HDL coding techniques is required, such as pipeline [52] and input and output registering, alongside advanced knowledge of the architectural resources of the device being used.

TABLE 2. Main FoMs (maximum number of bins 2^N, maximum number of counts $2^M - 1$, area occupancy, latency L, maximum clock frequency F_{CLK}, and maximum rate R) of state-of-the-art FPGA-based solutions.

Ref	N	M	LUTs(1)	FFs(1)	BRAM	L	F_{CLK} [MHz]	R [MSps]	FPGA Model
[36]	8	32	3400	6800	-	6	100	100	Altera Cyclone IV EP4CE22
[37]	8	22	15280	30560	-	-	280	280	unspecified
[11]	8	8	11850	9594	-	-	100	100	Xilinx Zynq XC7020 (28-nm)
[38]	8	14	218	213	5	-	200	50	Xilinx Artix-7 XC7A100T (28-nm)
[12]	16	8	3865	4903	-	-	370	370	Xilinx Zynq XC7Z030 (28-nm)
[46]	16	8	976	359	33	-	85	85	Xilinx Virtex II Pro
[47]	16	8	1265	1862	-	-	121	121	unspecified

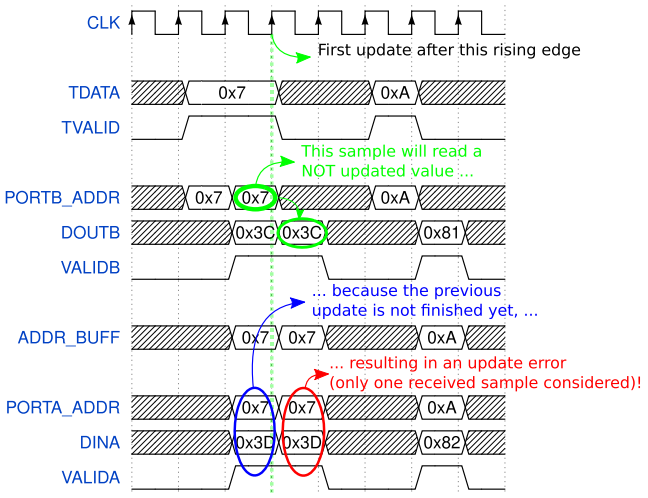


FIGURE 5. Timing diagrams highlighting the error condition that occurs in the presence of consecutive pieces of data of equal value, with the buffering technique only. Here, two pieces of data of equal values arrive in sequence and, due to the read/write latency, the read value for the second one (PORTB_ADDR = 0 × 7, DOUTB = 0 × 3C, green circled) is not yet updated, since the first update is not yet performed (PORTA_ADDR = 0 × 7, DINA = 0 × 3C+1 = 0 × 3D, blue circled). This leads to an error in the final update (PORTA_ADDR = 0 × 7, DINA = 0 × 3C+1 = 0 × 3D, red circled, instead of 0 × 3C+2 = 0 × 3E).

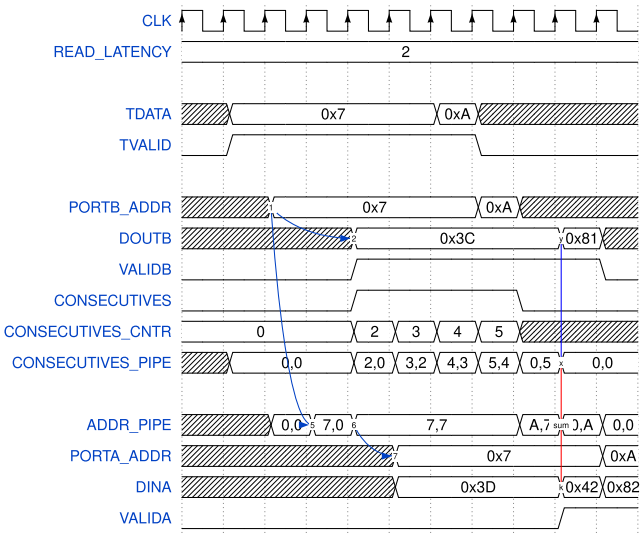


FIGURE 6. Timing diagrams of the consecutive management procedure in the case of five equal pieces of data in cascade. A read latency of two clock cycles is supposed. After the consecutive sequence of equal data ends, the correct overwrite of the stored content of bin elements is performed, that is 0 × 3D + 5 = 0 × 42. The timing sequence is described in the main text.

B. CONSECUTIVE MANAGEMENT

In the case of consecutive data with same value, this solution alone does not work. In fact, the read value of the bin content, requested when the first event comes, will not be updated yet when the second, equal, address arrives. This will lead to a wrong update process, as Figure 5 shows.

In order to solve this issue, a custom procedure, called ''consecutive management'', has been designed.

Consecutive management is intended for the general case of the consecutive arrival of equal data. In this regard, Figure 6 refers to the case of five events with a value of 0×7 arriving one after the other. In this eventuality, a flag asserting the event of consecutive data of same value is raised and stays raised as long as the incoming data keeps having the same value. In parallel, a counter keeps track of the number of these consecutive pieces of data. When the first one with a different value is received, the flag is de-asserted and the update operation is performed only once, on the basis of the counter's final value.

However, the use of the Macro provided by Xilinx introduces an additional issue, which is the intrinsic latency of the read operation in the XPM_MEMORY_SDPRAM module: for performance reasons, i.e. the maximization of data throughput, this latency is greater than one clock cycle, in order to also meet the timing constraint in the design. This is due to the implementation of internal pipeline stages. As a consequence, an architecture for managing the presence of this latency has to be put in place to process the incoming data at maximum rate.

C. LATENCY

The dead-time introduced in generating the histogram by the latency of the memory module operation scrambles the update operation. In fact, the address arrives ahead of the data value read from the memory location pointed out. This can be seen in Figure 7, where a latency of 2 clock cycles is supposed, and the contents of the requested addressed bins (DOUTB $= 0 \times 10, 0 \times 20, 0 \times 30$, green circled) are available one clock cycle later than the addresses used on the write port (ADDR_BUFF $=$ PORTA_ADDR $= 0 \times 1, 0 \times 2, 0 \times 3$, circled in red).

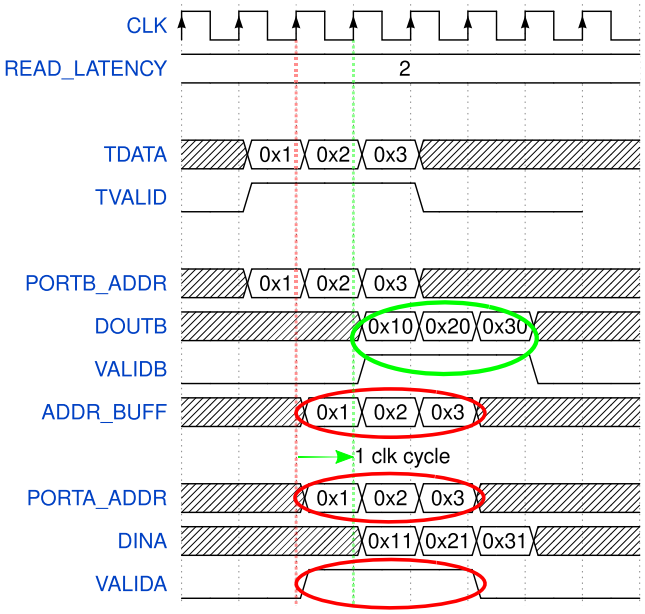


FIGURE 7. Wrong read/write operation of histogram update process, due to the presence of the latency. A read latency of two clock cycles is supposed.

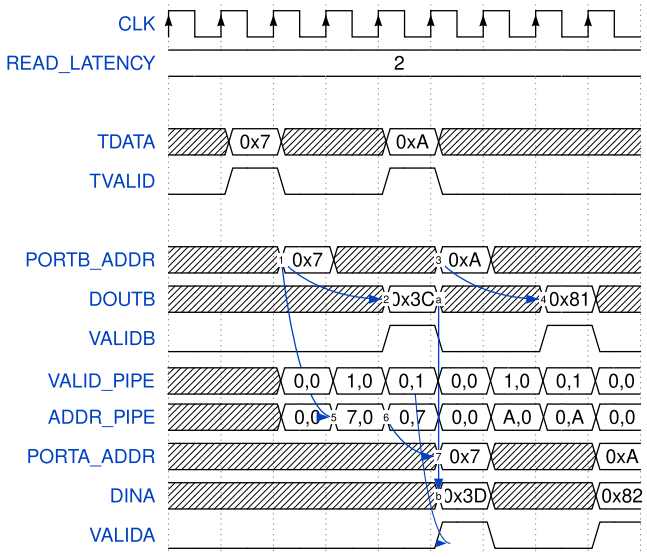


FIGURE 8. Read/write operations managed by means of a pipeline structure that ensures high rate performance despite the presence of memory latency. The timing sequence is described in the main text; a read latency of two clock cycles is considered. Here consecutive pieces of data are separated by at least two clock cycles.

In order to correctly update the histogram while maintaining the maximum speed rate, the write operation has to take place on the write port, Port-A, at the same address requested on the read port, Port-B, but only after the memory latency has passed, when the data becomes actually available. The coordination of read and write operations on the same address is mandatory. This issue, together with the management of consecutives, can be solved by means of a pipeline architecture, that involves the partitioning of the processing task into a larger number of steps: when applied sequentially, they will produce the same final results as the original single task, without having to drop the speed rate [52].

With reference to Figure 8, when a data arrives (TDATA $= 0 \times 7$), after the latency of the supposed two clock cycles, the content of the memory bin (DOUTB $= 0 \times 3C$) addressed by the TDATA value in PORTB_ADDR is available. PORTB_ADDR, in fact, simply contains the registered TDATA.

To get rid of the effect of this latency, a shift register (ADDR_PIPE) of length equal to the latency is used. Another shift register (VALID_PIPE), put in parallel, is fed a high value, corresponding to the data value entered in ADDR_PIPE. The usefulness of this second shift register lies in the management of the arrival of consecutive equal data, as will be illustrated below.

At the next clock cycle, when address 0×7 is in PORTB_ADDR, it enters the shift register ADDR_PIPE from the left (7, 0). After the two cycles of memory latency, ADDR_PIPE gives the correct address to the right (0,7), to the register PORTA_ADDR, and the incremented value ($0 \times 3C+1 = 0 \times 3D$) can be written in the correctly addressed bin (DINA) of the histogram.

The same mechanism is valid for all other [Address, Data with latency] pairs. In fact, the address received after 0×7 is 0xA, which is put in the pipeline at the next cycle (A,7). When the relative data is received in reading (0×81), the increment occurs (DINA $= 0 \times 81+1 = 0 \times 82$), with the relative writing, again at the address 0xA (PORTA_ADDR). A schematic view of the management of consecutive as well as the implemented pipeline is shown in Figure 9.

In reality, in a case like this, where the two consecutive pieces of data are spaced by, at least, the latency value, a pipeline mechanism would not be needed to keep the data value in memory when the corresponding bin value needs to be updated. For example, a simple buffer might suffice. However, to optimize implementation, this case is also handled by the pipeline mechanism which, instead, becomes indispensable when consecutive pieces of data arrive at a distance shorter than the latency. In the case illustrated in Figure 10, data arrives sequentially at each clock cycle. The mechanism is the same as that described in Figure 8.

The case of consecutive data of the same value, and at a distance below the latency, requires specific clarification.

Again we make use of a timing diagram, illustrated in Figure 11, to address this issue.

Every time a data value repeats at a distance shorter than the latency (for instance in Figure 10 the value 0×7 arrives again after 1 clock cycle), the pipelined read/write process fails. That is due to the fact that the read operation of the latest data value (PORTB_ADDR $= 0 \times 7$ circled in green in Figure 10), would not be counted in the corresponding histogram bin. In fact, the write operation (PORTA_ADDR $= 0 \times 7$, DINA $= 0 \times 3D$, orange circled) hasn't yet taken place when this second sample arrives. This leads to an error in the

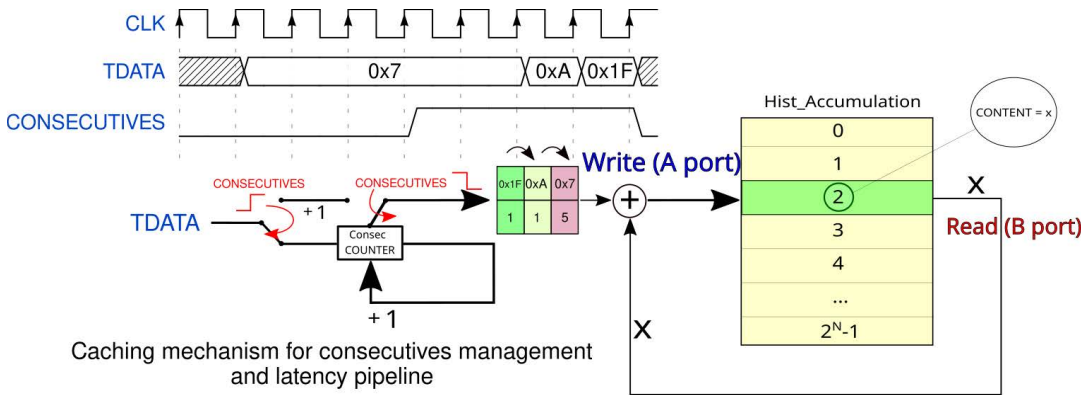


FIGURE 9. Simplified schematic view of the design with consecutive management and pipeline.

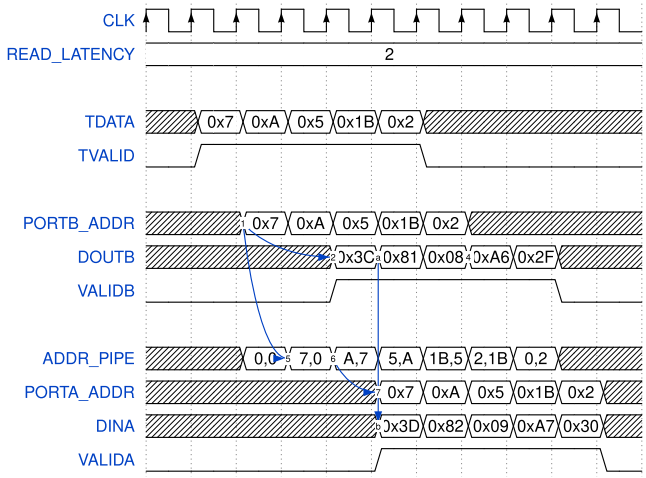


FIGURE 10. Read/write operations managed by means of a pipeline structure that ensures high rate performance despite the presence of memory latency. The timing sequence is described in the main text and a read latency of two clock cycles is supposed. Consecutive data are separated by less than two clock cycles and are different within a distance lower than the latency. The timing diagram of the VALID_PIPE shift register is not shown for the sake of clarity in the plot.

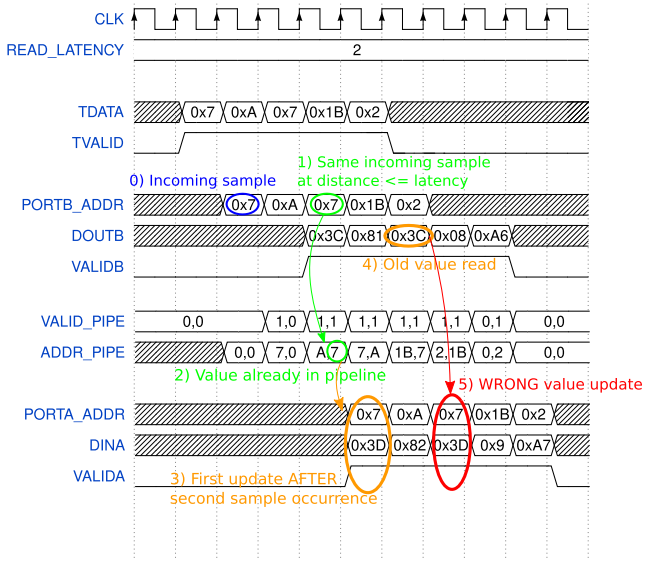


FIGURE 11. Failing of the pure pipeline mechanism. The timing sequence is described in the main text and a read latency of two clock cycles is supposed. Consecutive pieces of data also equal to each other are separated by less than two clock cycles.

write operation corresponding to the second sample (again PORTA_ADDR $= 0 \times 7$, DINA $= 0 \times 3$D, circled in red, instead of DINA $= 0 \times 3$E): the read value of a not yet updated address results in a wrong updating process, where one or more sample(s) is/are not counted as occurred events.

D. RE-PIPING

This problem was solved by modifying the structure of the pipeline presented, by introducing a technical novelty in the design of the histogrammer not previously found in the State-of-the-art. In this regard, we have called the modification introduced "re-piping".

Two new shift registers COUNTER_PIPE and VALID_PIPE have been introduced. With reference to Figure 12, as soon as an incoming sample is found to be already present in the pipeline (PORTB_ADDR $= 0 \times 7$, ADDR_PIPE[1] $= 0 \times 7$, circled in red), the corresponding value in COUNTER_PIPE is increased by 1 (COUNTER_PIPE[0] $= 2$, circled in blue). In the same clock cycle the update related to the first sample should be accomplished, but by lowering the VALIDA signal of Port-A (VALIDA $= 0$, circled in blue) the write operation is avoided. The update is postponed, with the correct increase equal to the value in COUNTER_PIPE (PORTA_ADDR $= 0 \times 7$, DINA $= 0 \times 3$C+2 $= 0 \times 3$E, VALIDA $= 1$, circled in green).

A graphical view of the mechanism is shown in Figure 13.

E. SWAPPING

Of course, once the histogram is built, it must be available to be read by surrounding processing units.

However, it may happen that, while the histogram is being read, data arrives which must be classified in the histogram too. The possible contemporaneity of these two events needs to be managed.

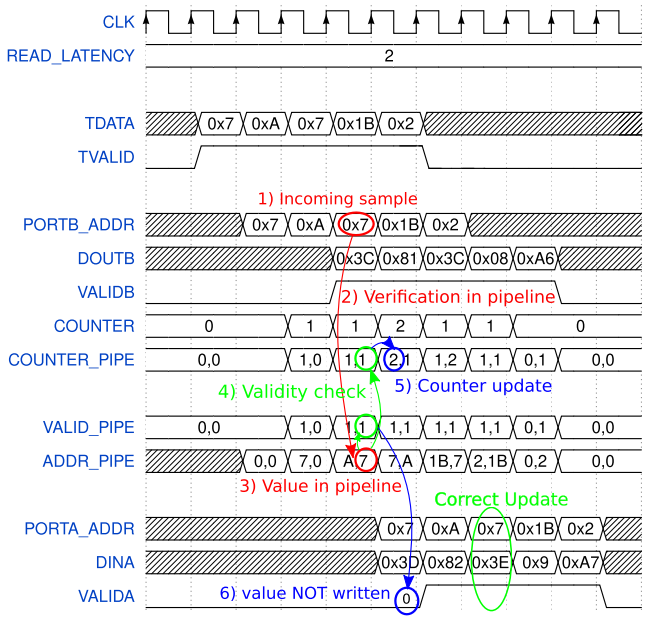


FIGURE 12. Pipelined structure managing the read/write operation with the implemented re-piping feature. The timing sequence is described in the main text and a read latency of two clock cycles is supposed.

The solution implemented was to instantiate two SDPRAM memories in parallel, making them work as described here: the histogram is constructed in a memory unit until an overall number of classified values, set a priori, is reached. At this point, the data at the input of the histogram is redirected to the other implemented memory, where the construction of a new histogram begins, ending again once the maximum preset value of samples is reached. While integrating the new histogram on the second memory unit, the first one is made available for reading. This process is continuously deployed, making it possible to always have one histogram being built and the previous one available for readout. The mechanism, illustrated in Figure 14, is based on a counter which, upon reaching the maximum set number of samples in the histogram, asserts a flag which commands switching of the data input from one SDPRAM memory to another.

This swapping technique therefore allows simultaneous reading of the histogram and classification of all incoming data without interruption.

The read-out of the completed histogram is done via an Advanced eXtensible Interface 4 (AXI4) slave interface. To implement this, the read-out module takes advantage, as read time, of the integration time of the other replica (Figure 14). If, during the reading operation, an overflow takes place in one bin of the histogram under construction, an error flag is asserted.

F. HARDWARE RESULTS

The histogram processor described has been implemented in FPGA, with user-programmable histogram dimensions in terms both of bin width (M) and of number of bins (N), and is compatible with the 7-Series device family from Xilinx. Thanks to the proposed design a latency equal to 2 clock cycles (i.e., $L = 2$) and a maximum rate equal to the clock frequency (i.e., $R = F_{CLK}$) are guaranteed independently from N and M.

Of course, absolute and relative area occupancy and the operating clock frequency of the implemented architecture depend on the target device selected, and on N and M. In this sense, Table 3 refers to area occupancy and achieved performance on a FPGA Xilinx Artix-7 XC7A35T (20.8k LUTs and 41.7k FFs) hosted on a Digilent Basys3 board, that is actually one of the smallest and slowest FPGA of 7-Series devices [53]. This choice was made to highlight the efficiency of the proposed solution, both in terms of resource Utilization and in guaranteed operating frequencies. It must be underlined that the number of bins of the histograms in the table correspond to 2^N, that every bin can count up to $2^M - 1$ events, and that the target measurement rate of 200 MSps is always guaranteed.

A summarized, 3D graphical representation of the obtained Frequency and Area occupation parameters, as function of N and M values, can be seen in Figure 16: in Figure 16a is shown the percent area occupation over the "Utilization %" columns of Table 3; while, Figure 16b represents the percent frequency gain calculated with respect to the average of the

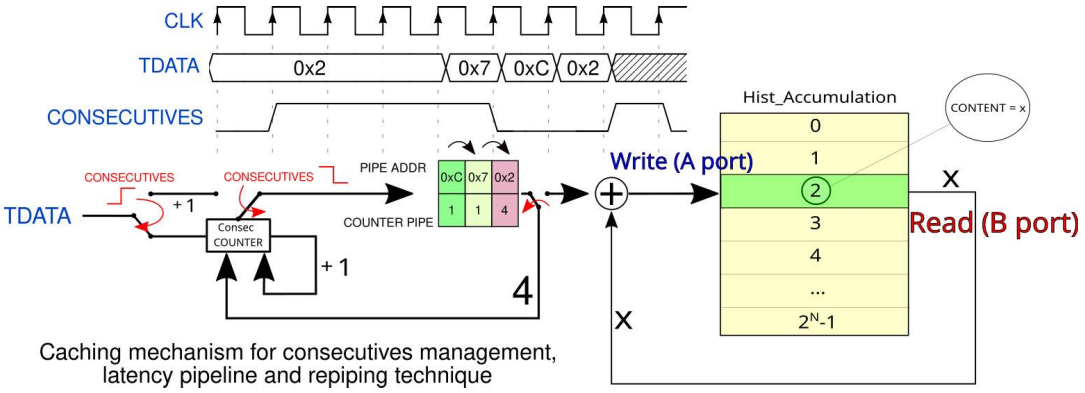
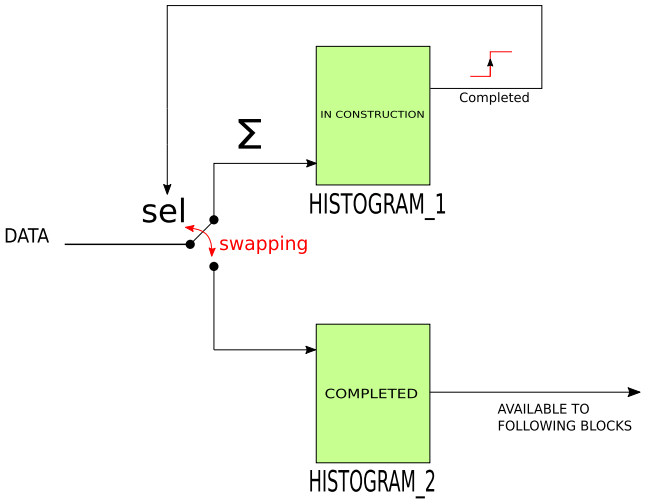
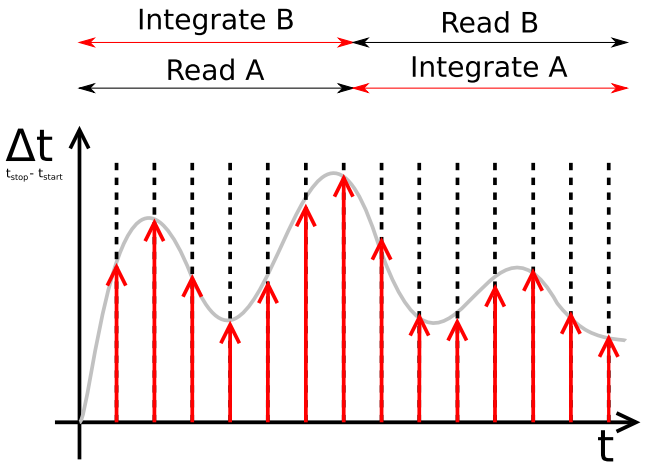


FIGURE 13. Simplified schematic view of the design with the Re-Piping technique upgrade.



(a) Swapping technique schematic representation.



(b) Concept of integration-read processes using swapping technique

FIGURE 14. Swapping technique.

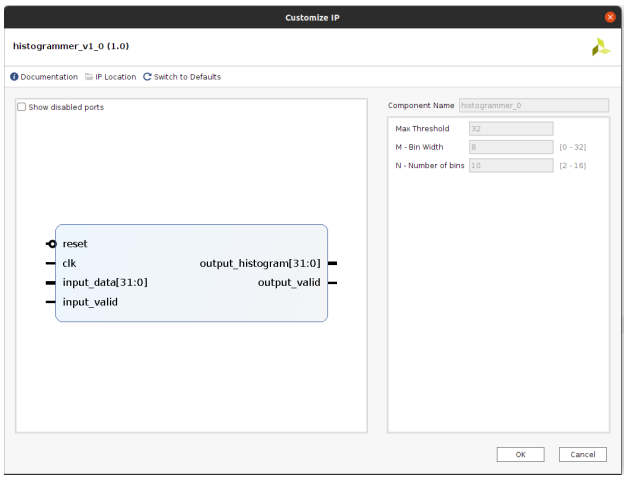


FIGURE 15. Generic histogram IP Core, with tunable M, N and threshold for the maximum bin value.

clock frequencies (i.e., $F_{AVG} = (100+280+100+200+370+85+121)/7 \cong 180\,MHz$) of Table 2, i.e. $(F_{CLK} - F_{AVG})/F_{AVG}$ (where F_{CLK} is the "Frequency" column of Table 3).

Table 3 allows a comparison between the proposed solution and the presented state-of-the-art, which is reported in 2, with N and M used as tuning parameters. It has, however, to be noted that the proposed solution performance would increase by a noticeable margin by simply using a higher speed-grade device of the same family (e.g. Artix-7 -2 instead of Artix-7 -1 increases the frequency of a factor ≅1.2) or a higher-end FPGA (e.g. Kintex-7 instead of Artix-7 increases the frequency of a factor ≅1.18 at a same speed-grade) and relative area occupation would, of course, decrease using a larger device (e.g. A100T with respect to a A35T reduces the percent area occupancy by a factor ≅2.86).

G. PORTABILITY

The very same considerations apply also in the case of a migration to a more recent generation of FPGA devices (i.e. UltraScale and UltraScale+ with respect to 7 Series) or even to different manufacturers (i.e. Intel or Lattice), with minor efforts that would consist of just replacing the primitive used for the memory instantiation.

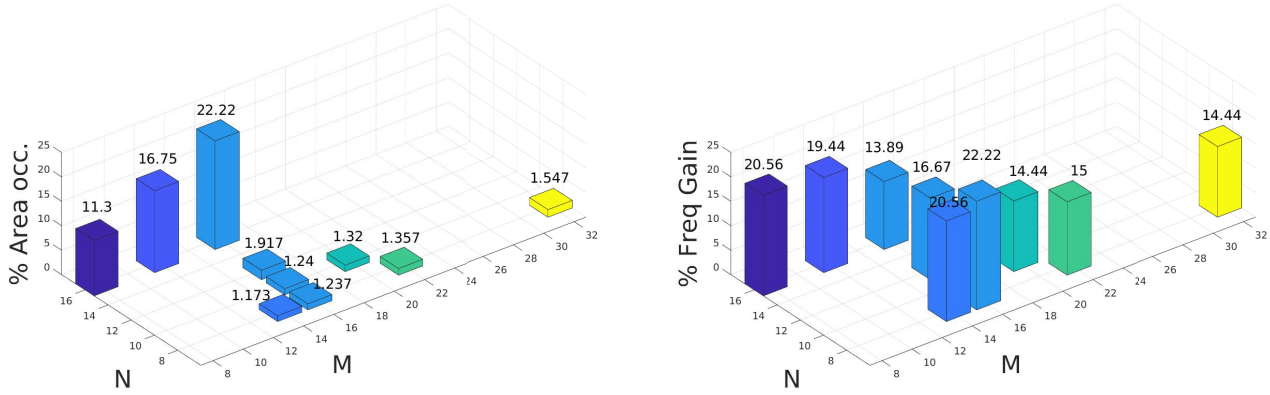
Concerning the case of Xilinx newer devices the migration is effortless, as the Macro available for the 7 Series family (Simple Dual Port Ram), which is used in our design, is the same across the last 3 generations of FPGAs. As regards, instead, the migration to different manufacturers, a comparison between the primitives by Xilinx, Intel and Lattice, the three largest FPGA manufacturers, is shown in Figure 17.

As can be seen, there is a direct correlation between the majority of the signals among the primitives: Xilinx' "dina", "addra" "ena" and "clka" correspond to Intel's "data", "wraddress", "wren" and "wrclk" and Lattice's "wr_data_i", "wr_addr_i", "wr_en_i" and wr_clk_i". This correlation directly translates into a completely similar behavior and properties of the primitives, that may, at most, differ from one another in terms of latency in the read/write operations. This is, however, totally manageable by the described IP-Core to cope with the required latency. All the primitives feature a read port composed of address, enable, data and clock signals as well, and other optional control signals. This example is to underline how the developed IP-Core solution is highly portable with minor efforts in the substitution of the utilized memory primitive.

IV. EXPERIMENTAL VALIDATION

All the tests were carried out to validate the correct functioning of the proposed histogram generator, above all by validating the critical issues solved by the pipeline(Paragraph III-C) and re-piping (Paragraph III-D) structure, as well as the continuous computation that the system is able to catalog thanks to the swapping architecture (Paragraph III-E).

The case study for the validation of the proposed architecture was chosen to be the calibration section of a high-performance Time-to-Digital Converter (TDC), of which in subsection IV-B. The most interesting aspect of this test



(a) Percent area occupation on the device.

(b) Percent frequency gain with respect to the average frequencies in the State-of-the-art (Table 2).

FIGURE 16. 3D graphical representation of Table 3 as function of N and M values.

TABLE 3. Features of implementation costs and performance of the proposed architecture for making histograms in a very low profile FPGA device (Xilinx Artix-7 XC7A35T).

(a) Implementation at 256 bins ($N = 8$) with 16383 counts per bin ($M = 14$).

Resource	Utilization	Utilization %	Frequency
LUT	193	0.93	
Flip Flops	246	0.59	217MHz
BRAM	1	2	

(b) Implementation at 256 bins ($N = 8$) with 65535 counts per bin ($M = 16$).

Resource	Utilization	Utilization %	Frequency
LUT	221	1.06	
Flip Flops	270	0.65	220MHz
BRAM	1	2	

(c) Implementation at 256 bins ($N = 8$) with 4194303 counts per bin ($M = 22$).

Resource	Utilization	Utilization %	Frequency
LUT	259	1.25	
Flip Flops	342	0.82	207MHz
BRAM	1	2	

(d) Implementation at 256 bins ($N = 8$) with 4294967295 counts per bin ($M = 32$).

Resource	Utilization	Utilization %	Frequency
LUT	320	1.54	
Flip Flops	459	1.10	206MHz
BRAM	1	2	

(e) Implementation at 1024 bins ($N = 10$) with 65535 counts per bin ($M = 16$).

Resource	Utilization	Utilization %	Frequency
LUT	217	1.04	
Flip Flops	284	0.68	207MHz
BRAM	1	2	

(f) Implementation at 1024 bins ($N = 10$) with 1048575 counts per bin ($M = 20$).

Resource	Utilization	Utilization %	Frequency
LUT	242	1.16	
Flip Flops	332	0.80	206MHz
BRAM	1	2	

(g) Implementation at 4096 bins ($N = 12$) with 65535 counts per bin ($M = 16$).

Resource	Utilization	Utilization %	Frequency
LUT	214	1.03	
Flip Flops	298	0.72	210MHz
BRAM	2	4	

(h) Implementation at 256*256 bins ($N = 16$) with 255 counts per bin ($M = 8$).

Resource	Utilization	Utilization %	Frequency
LUT	297	1.43	
Flip Flops	201	0.48	217MHz
BRAM	16	32	

(i) Implementation at 256*256 bins ($N = 16$) with 4095 counts per bin ($M = 12$).

Resource	Utilization	Utilization %	Frequency
LUT	352	1.69	
Flip Flops	237	0.57	215MHz
BRAM	24	48	

(j) Implementation at 256*256 bins ($N = 16$) with 65535 counts per bin ($M = 16$).

Resource	Utilization	Utilization %	Frequency
LUT	412	1.99	
Flip Flops	273	0.66	205MHz
BRAM	32	64	

TABLE 4. Comparison of the proposed IP-Core with *N* and *M* tuned at the value proposed in the state-of-the-art in Table 3.

Ref	N	M	LUTs	FFs	BRAM	L	F_{CLK} [MHz]	R [MSps]	FPGA Model
[36]	8	32	3400	6800	-	6	100	100	Altera Cyclone IV EP4CE22
Proposed IP-Core	8	32	320	459	1	2	206	206	Artix-7 XC7A35T
[37]	8	22	15280	30560	-	-	280	280	unspecified
Proposed IP-Core	8	22	259	342	1	2	207	207	Artix-7 XC7A35T
[11]	8	8	11850	9594	-	-	100	100	Xilinx Zynq XC7020 (28-nm)
Proposed IP-Core	8	32	320	459	1	2	206	206	Artix-7 XC7A35T
[38]	8	14	218	213	5	-	200	50	Xilinx Artix-7 XC7A100T (28-nm)
Proposed IP-Core	8	14	193	246	1	2	217	217	Artix-7 XC7A35T
[12]	16	8	3865	4903	-	-	370	370	Xilinx Zynq XC7Z030 (28-nm)
[46]	16	8	976	359	33	-	85	85	Xilinx Virtex II Pro
[47]	16	8	1265	1862	-	-	121	121	unspecified
Proposed IP-Core	16	8	297	201	16	2	217	217	Artix-7 XC7A35T

environment is the fact that the TDC considered is a complex system, featuring multi-channel capabilities (16 channels) and performing at a high-rate (100 MSps) [54], and perfectly adapts to the features that are the object of the design of the proposed histogram architecture, which are high-speed, low-latency and high number of channels. Moreover, the same histogramming architecture is used for a second scope in the same case study, as described in IV-C: it is employed as a processing tool to build statistics of the measurements taken on different channels of the same TDC system, providing first order insights about the physical processes that gave rise to the measurements. This double purpose with which the histogramming architecture is involved in our case study is proof of the great flexibility of the proposed solution.

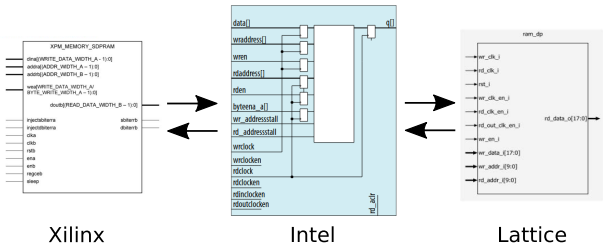


FIGURE 17. Representation of the dual port RAM primitives by different manufacturers.

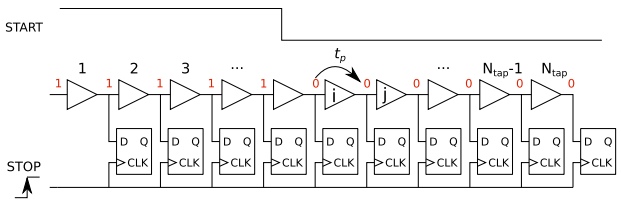


FIGURE 18. Structure of the delay line that is the basis of the TDC, sampling an input signal.

A. TDC

A TDC is a digital electronic measurement system that converts the time distance between two events, or one event and a reference, into a digital code. The events limiting the time interval being measured are usually referred as START and STOP, and the measurement is generally called a timestamp.

The considered TDC is a Tapped Delay-Line (TDL) TDC [55] (Figure 18), that is made up of a chain of N_{Tap} buffers (a.k.a. taps), i.e. Delay-Line (DL), characterized by a proper propagation delay t_p, the outputs from which are registered by D Flip-Flops (DFFs). The START event is propagated over the DL and when the STOP event occurs, the DL is sampled by the DFFs. In this manner, the measurement of the time interval consists of counting the number of buffers, a.k.a. bins, that have registered a high value (i.e. '1').

The measurement resolution of this architecture is the propagation delay t_p of each buffer, and the maximum measurable interval comes from the total number of chained buffers.

Ideally, the real values of the propagation delays of the buffer should all be known exactly.Unfortunately, the presence of high Process Voltage and Temperature (PVT) fluctuations leads to big mismatches in the values of the propagation delays t_p between taps, up to a factor of 10 compared to their mean value [56]: the $i-th$ tap is characterized by a propagation delay $t_p[i]$ that is different to the propagation delay of the tap $j-th$ $t_p[j]$. In this sense, the presence of real-time calibration system that computes the histogram as a kernel is required [57].

In real world applications, TDCs are used in time-based experiments. The purpose of a TDC is to measure time intervals, which represent the information to be extracted by the experiment; following the timestamp generation, histograms are used to analyze the statistics of the observed phenomena and, thus, derive information by means of computation of the statistical moments. In the case of TCSPC, for example, the TDC measures the time a sample being tested needs to emit photons when stimulated by an external source; in this context the histogram of these delays is used to characterize the spectral properties of the material. In TOF experiments, instead, (e.g. LR, TOF-PET, TOF-MS) the TDC is in charge of measuring the TOF of the particles and the histograms

are used to extract the mean values and the standard deviation, for instance, in order to identify the spatial position (e.g. LR, TOF_PET) or the mass (e.g. TOF-MS); hence, the requirement of using FPGA-based histograms as hardware accelerators for real-time application is mandatory.

In Paragraph IV-B, we present the proposed histogram IP-Core for TDC real-time calibration, while in Paragraph IV-C, we use the proposed solution to plot the histogram of the measured timestamps for time-based experiments in real-time.

B. TDC CALIBRATOR

The calibration process needs to address each single delay of the buffers that compose the DL. In order to do so, a statistical approach is used: if the repetitively sent START and STOP signals on the DL belong to a white distribution and are uncorrelated, ranging from distance 0 to FSR ($FSR = \sum_{n=0}^{N_{Tap}} t_p[n]$) from one another, by building a histogram of the statistics of the bins that are hit and subsequently normalizing it, an estimation of the propagation times $t_p[n]$ with $n \in [0; N_{Tap} - 1]$ is obtained, and the table that contains the estimation of tap values is called Calibration Table ($CT[n]$ with $n \in [0; N_{Tap} - 1]$). In this context:

$$CT[n] = \frac{h[n]}{K} \cdot FSR$$

where $h[n]$ represents the count accumulated on the $n-th$ buffer of the histogram and K is total number of histogrammed samples, i.e. $\sum_n h[n] = K$. This leads to:

$$t_p[n] = CT[n] + o\left(\frac{FSR}{K}\right)$$

which is the desired estimation of the propagation time of each buffer with an error of up to FSR/K.

For the sake of simplicity, the histogramming architecture used for the calibration of a TDC channel will be referred to as a ''calibrator''. The calibration parameters directly affect the hardware resources needed to perform the computation; in fact, as the histograms representing the CT are integrated into the BRAM resources of the FPGA on which the TDC is implemented, their dimensions are inferred from the calibration needs.

The first thing to consider is the number of channels that need to be calibrated: this number translates directly into the number of replicas of the calibrators that the design will need. As a consequence, the higher the number of channels to calibrate, the higher the number of instances of the calibrator. Moreover, each instance's resource usage depends on the dimension of the CT required for calibration: the longer the DL to be characterized, the higher the number of memory locations needed to fit it (each bin corresponds to one address location), while the more precise a calibration needs to be, the higher the number of samples for the construction of the statistics will be, leading to a larger size for each bin of the histogram. Referring to Paragraph III-F, it can be concluded that the dimension of the memory (D) needed by a single instance of the calibrator is: $D = 2^N \cdot M$, where $N_{Tap} = 2^N$ and M are chosen in order to have an negligible error in the CT estimation, i.e. $2^M - 1 > K$. Considering a TDC multi-channel solution at N_{CH} channels, the total memory usage for the calibration is $D_{tot} = D \cdot N_{CH}$.

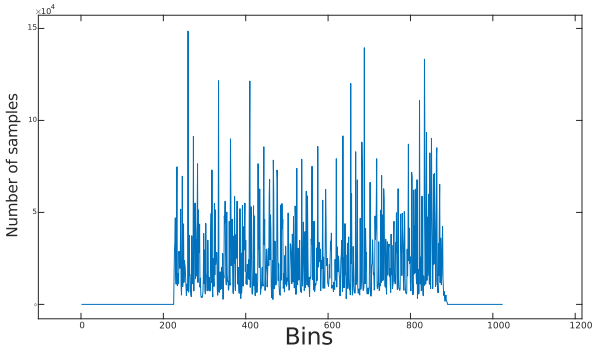


FIGURE 19. Calibration table of a single TDC channel, performed by the histogram core and used to calibrate the tapped-delay line.

However, in terms of performance, the histogrammer's architecture has been developed using the techniques described in Section III, in order to achieve timing closure at 200 MHz, which is a frequency able to cope with the TDC operating frequencies, independently of the dimensions of the parameters involved, up to the required maximum of $N = 12$ for the addressing part and $M = 20$ for the bin width.

We wanted to explore the worst case by proceeding to implement the calibrator based on the proposed histogram architecture into the TDC structure, on a FPGA Artix-7 XC7A35T (20.8k LUTs and 41.7k FFs) that is actually one of the smallest and slowest available Xilinx device of the 7-Series family. The development board used in this test was a Digilent Basys3. The assessments of the area used and performance achieved, reported in Table 5 and derived from a 16-channel TDC firmware version, after the placement and routing stage and bitstream generation, highlight the efficiency of the proposed architecture for different implementation features at different N_{Taps}, i.e. 256, 1024, 4096, and M, i.e. 16, 20.

The implementation of the architecture in a multi-channel version of the TDC was also carried out, noting no worsening of operational performance against a linear increase, in first order approximation, of the necessary resources involved.

A representation of a Calibration Table referred to the a single channel of the TDC, calculated by the presented histogram core and acquired at a rate of 50 Msps, with a minimum dead-time of 5 ns between the events, is shown in Figure 19.

C. TIMESTAMP HISTOGRAM

For the timestamp histogramming process, the IP-Core designed in Section III computes the histogram of the time difference between the timestamps of the events observed on two different channels of the TDC in real-time.

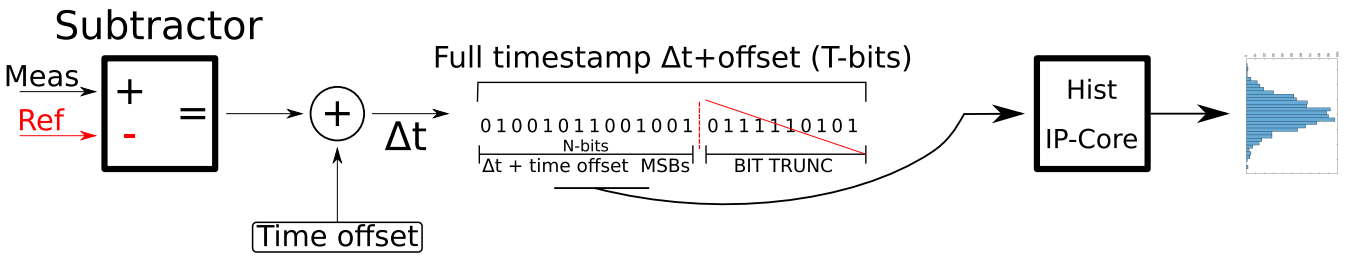


FIGURE 20. Graphical representation of the *N* bits of Δt stored into the histogram.

TABLE 5. Implementation costs and performance of a single calibrator instance realized with the proposed architecture, derived from a 16-channel TDC implementation on a low profile 7 series Xilinx FPGA device. The BRAM occupation is expressed in terms of number of 36Kb RAM blocks (each one made up of two independent 18Kb blocks) which are peculiar of the 7 series FPGAs.

(a) Implementation at $N_{Taps} = 256$ ($N = 8$) up to 65535 counts per tap ($M = 16$).

Resource	Utilization Absolute	%	Frequency
LUT	273	1.31	
Flip Flops	322	0.77	224 MHz
BRAM	1.5	3	

(b) Implementation at $N_{Taps} = 1024$ ($N = 10$) up to 65535 counts per tap ($M = 16$).

Resource	Utilization Absolute	%	Frequency
LUT	293	1.41	
Flip Flops	337	0.81	217 MHz
BRAM	3	6	

(c) Implementation at $N_{Taps} = 4096$ ($N = 12$) up to 65535 counts per tap ($M = 16$).

Resource	Utilization Absolute	%	Frequency
LUT	301	1.45	
Flip Flops	350	0.84	201 MHz
BRAM	6	12	

(d) Implementation at $N_{Taps} = 1024$ ($N = 10$) up to 1048575 counts per tap ($M = 20$).

Resource	Utilization Absolute	%	Frequency
LUT	345	1.66	
Flip Flops	389	0.94	210 MHz
BRAM	3	6	

In this sense, a subtractor module between the TDC and the histogrammer is mandatory. The subtractor selects which channel of the TDC is used as a reference (CH_{REF}) and which as a measurement (CH_{MEAS}). In this way it can output the time difference ($\Delta t = t_{MEAS} - t_{REF}$) between the timestamps generated by CH_{MEAS} (t_{MEAS}) and CH_{REF} (t_{REF}).

Moreover, in order to to adapt the dimension of Δt, that is $T - bit$ wide, to the 2^N bins offered by the histogram, two registers called TIME_OFFSET and BIT_TRUNC have been provided, as reported in Figure 20. In this manner, not all of the T bit of Δt, but only those from BIT_TRUNC to $N +$ *BIT_TRUNC* of $\Delta t +$ *TIME_OFFSET* are histogrammed. A graphical representation showing time offset and the histogrammed window is visible in Figure 21. Lastly, as one bin of the histogram reaches a value close to $2^M - 1$, the module asserts a proper flag for signaling bin overflow.

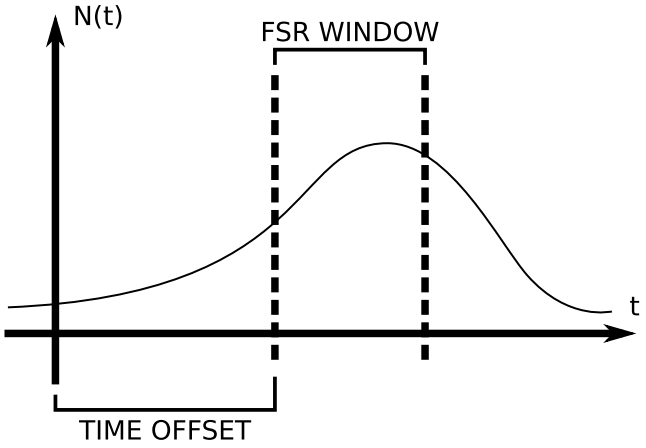


FIGURE 21. Graphical representation of time offset and histogrammed window.

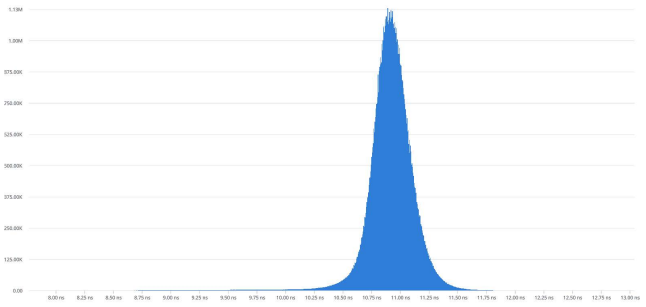


FIGURE 22. TDC acquisition of a SiPM-based optical spectroscopy application by means of the histogram core.

The ability to build histograms of timestamps is a great value, as previously mentioned, in applications such as TCSPC, TOF-PET, Optical Spectroscopy and many others. We report, in Figure 22, an acquisition from an Optical Spectroscopy setup, where a SiPM is read out by the aforementioned TDC, which exploits the hardware acceleration offered by the histogram core to elaborate incoming measurements

at 40 Msps, with a minimum dead-time of 10 ns between the events.

V. CONCLUSION AND FUTURE DEVELOPMENTS

To conclude, an architecture for the computation of histograms with high efficiency and performance, using an innovative technique and suited for programmable logic has been presented.

After an overview of the state-of-the-art in terms of architectures for the computation of histograms, showing the advantage of parallel computing strategies based on GPUs and FPGAs compared to temporal computing ones, an FPGA-based, application agnostic architecture focusing on efficiency and performance has been proposed.

Implementation issues and a state-of-the-art area-saving solution are proposed as an IP-Core, compatible with multichannel applications independently both of the scientific field and of the device in use, thanks to a high cross-manufacturer compatibility.

Moreover, experimental evaluations using an existing multichannel FPGA-based TDC were performed. In detail, the proposed histogram structure is used both to calibrate the TDC in real-time with a measurement rate of up to 224 MSps and to extract statistics on the measures of the TDC at the same high rate.

Future upgrades will be developed in order to allow the implementation of the histogrammer IP Core with variable latency values, which are now limited to 2 clock cycles from the GUI, in order to cope with any memory primitive in use.

Moreover, in order to permit even greater flexibility, we will enable the possibility to choose whether or not to enable the swapping mechanism, which is now always active; this will make it possible to implement the histogrammer with even lower BRAM occupation, trading it off with the contemporary availability of the update mechanism and the histogram readout mechanism.

REFERENCES

[1] A. Gundlach-Graham, L. Hendriks, K. Mehrabi, and D. Günther, "Monte Carlo simulation of low-count signals in time-of-flight mass spectrometry and its application to single-particle detection," *Anal. Chem.*, vol. 90, no. 20, pp. 11847–11855, Oct. 2018.

[2] S. Dutta, A. Abhinav, P. Dutta, P. Kumar, and A. Halder, "An efficient image compression algorithm based on histogram based block optimization and arithmetic coding," *Int. J. Comput. Theory Eng.*, vol. 4, no. 6, p. 954, 2012.

[3] R. Kapoor, R. Gupta, L. H. Son, S. Jha, and R. Kumar, "Detection of power quality event using histogram of oriented gradients and support vector machine," *Measurement*, vol. 120, pp. 52–75, May 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0263224118300940

[4] Z. Istvan, L. Woods, and G. Alonso, "Histograms as a side effect of data movement for big data," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, Jun. 2014, pp. 1567–1578.

[5] Y. Peng, M. C. Fu, B. Heidergott, and H. Lam, "Maximum likelihood estimation by Monte Carlo simulation: Toward data-driven stochastic modeling," *Oper. Res.*, vol. 68, no. 6, pp. 1896–1912, Nov. 2020.

[6] H. Pereira and R. C. Marques, "An analytical review of irrigation efficiency measured using deterministic and stochastic models," *Agricult. Water Manage.*, vol. 184, pp. 28–35, Apr. 2017.

[7] T. N. Palmer, "Stochastic weather and climate models," *Nature Rev. Phys.*, vol. 1, no. 7, pp. 463–471, Jul. 2019.

[8] Y. Han and S. Ahn, "Stochastic modeling of breakdown at freeway merge bottleneck and traffic control method using connected automated vehicle," *Transp. Res. B, Methodol.*, vol. 107, pp. 146–166, Jan. 2018.

[9] W. Yang, S. Tang, M. Li, Y. Chen, and Z. Zhou, "Steganalysis of low embedding rates LSB speech based on histogram moments in frequency domain," *Chin. J. Electron.*, vol. 26, no. 6, pp. 1254–1260, Nov. 2017.

[10] R. Lima and R. Sampaio, "Parametric analysis of the statistical model of the stick-slip process," *J. Sound Vib.*, vol. 397, pp. 141–151, Jun. 2017.

[11] T. Bonny, T. Rabie, and A. H. A. Hafez, "Multiple histogram-based face recognition with high speed FPGA implementation," *Multimedia Tools Appl.*, vol. 77, no. 18, pp. 24269–24288, Sep. 2018, doi: 10.1007/s11042-018-5647-8.

[12] S. Hazra, S. Ghosh, S. P. Maity, and H. Rahaman, "A new FPGA and programmable SoC based VLSI architecture for histogram generation of grayscale images for image processing applications," *Proc. Comput. Sci.*, vol. 93, pp. 139–145, Jan. 2016, [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050916314338

[13] J. J. Hamill, "2D energy histograms for scatter estimation in an SiPM PET scanner," in *Proc. IEEE Nucl. Sci. Symp. Med. Imag. Conf. (NSS/MIC)*, Oct. 2019, pp. 1–4.

[14] M. Alayed and M. Deen, "Time-resolved diffuse optical spectroscopy and imaging using solid-state detectors: Characteristics, present status, and research challenges," *Sensors*, vol. 17, no. 9, p. 2115, Sep. 2017.

[15] J. Bouchard, A. Samson, W. Lemaire, C. Paulin, J.-F. Pratte, Y. Bérubé-Lauzière, and R. Fontaine, "A low-cost time-correlated single photon counting system for multiview time-domain diffuse optical tomography," *IEEE Trans. Instrum. Meas.*, vol. 66, no. 10, pp. 2505–2515, Oct. 2017.

[16] O. M. Mozos, H. Mizutani, H. Jung, R. Kurazume, and T. Hasegawa, "Categorization of indoor places by combining local binary pattern histograms of range and reflectance data from laser range finders," *Adv. Robot.*, vol. 27, no. 18, pp. 1455–1464, Dec. 2013.

[17] E. Salahat, H. Saleh, M. S. Zitouni, A. S. Sluzek, B. Mohammad, M. Al-Qutayri, and M. Ismail, "A robust histogram-based image segmentation ASIC design for system-on-chip using 65 nm technology," in *Proc. Int. Conf. Commun., Signal Process., their Appl. (ICCSPA)*, Feb. 2015, pp. 1–4.

[18] W. Kehl, F. Tombari, S. Ilic, and N. Navab, "Real-time 3D model tracking in color and depth on a single CPU core," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 745–753.

[19] M. Poostchi, K. Palaniappan, D. Li, M. Becchi, F. Bunyak, and G. Seetharaman, "Fast integral histogram computations on GPU for real-time video analytics," 2017, *arXiv:1711.01919*.

[20] E. Wang, L. Shi, J. Niu, Y. Hua, H. Li, X. Zhu, C. Xie, and T. Ye, "Multichannel spatially nonhomogeneous focused vector vortex beams for quantum experiments," *Adv. Opt. Mater.*, vol. 7, no. 8, Apr. 2019, Art. no. 1801415.

[21] C. L. Ruiz, D. L. Fehl, G. A. Chandler, G. Cooper, B. Jones, J. D. Styron, and J. Torres, "Multichannel, triaxial, neutron time-of-flight diagnostic for experiments at the *Z* facility," *Phys. Rev. Accel. Beams*, vol. 23, no. 2, Feb. 2020, Art. no. 020401.

[22] T. Ochiai, S. Watanabe, T. Hori, and J. R. Hershey, "Multichannel end-to-end speech recognition," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 2632–2641.

[23] S. M. Griffin, K. Inzani, T. Trickle, Z. Zhang, and K. M. Zurek, "Multichannel direct detection of light dark matter: Target comparison," *Phys. Rev. D, Part. Fields*, vol. 101, no. 5, Mar. 2020, Art. no. 055004.

[24] H. Choe, S. Gorfman, S. Heidbrink, U. Pietsch, M. Vogt, J. Winter, and M. Ziolkowski, "Multichannel FPGA-based data-acquisition-system for time-resolved synchrotron radiation experiments," *IEEE Trans. Nucl. Sci.*, vol. 64, no. 6, pp. 1320–1326, Jun. 2017.

[25] W. Fu, B. Wei, X. Li, Q. Wang, and X. Hu, "A low delay transmission method of multi-channel video based on FPGA," *IOP Conf. Ser., Mater. Sci. Eng.*, vol. 322, no. 5, 2018, Art. no. 052032.

[26] D. Shi, W.-S. Gan, J. He, and B. Lam, "Practical implementation of multichannel filtered-x least mean square algorithm based on the multiple-parallel-branch with folding architecture for large-scale active noise control," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 28, no. 4, pp. 940–953, Apr. 2020.

[27] M. Baklouti, P. Marquet, J. L. Dekeyser, and M. Abid, "FPGA-based many-core system-on-chip design," *Microprocessors Microsyst.*, vol. 39, nos. 4–5, pp. 302–312, Jun. 2015.

[28] A. Al-Mahmood and M. Opoku, "A study of FPGA-based system-on-chip designs for real-time industrial application," *Int. J. Comput. Appl.*, vol. 163, no. 6, pp. 9–19, Apr. 2017.

[29] Xilinx. (2021). *Zynq-7000 SoC*. [Online]. Available: https://www.xilinx.com/support/documentation/user_guides/ug585-Zynq-7000-TRM.pdf

[30] A. Anand, V. Jha, and L. Sharma, "An improved local binary patterns histograms techniques for face recognition for real time application," *Int. J. Recent Technol. Eng.*, vol. 8, no. 2S7, pp. 524–529, 2019.

[31] T. Surasak, I. Takahiro, C.-H. Cheng, C.-E. Wang, and P.-Y. Sheng, "Histogram of oriented gradients for human detection in video," in *Proc. 5th Int. Conf. Bus. Ind. Res. (ICBIR)*, May 2018, pp. 172–176.

[32] F. M. D. Rocca, H. Mai, S. W. Hutchings, T. A. Abbas, K. Buckbee, A. Tsiamis, P. Lomax, I. Gyongy, N. A. W. Dutton, and R. K. Henderson, "A 128 × 128 SPAD motion-triggered time-of-flight image sensor with in-pixel histogram and column-parallel vision processor," *IEEE J. Solid-State Circuits*, vol. 55, no. 7, pp. 1762–1775, Jul. 2020.

[33] T. Okino, S. Yamada, Y. Sakata, S. Kasuga, M. Takemoto, Y. Nose, H. Koshida, M. Tamaru, Y. Sugiura, S. Saito, S. Koyama, M. Mori, Y. Hirose, M. Sawada, A. Odagawa, and T. Tanaka, "A 1200×900 6μm 450 fps geiger-mode vertical avalanche photodiodes CMOS image sensor for a 250 m time-of-flight ranging system using direct-indirect-mixed frame synthesis with configurable-depth-resolution down to 10 cm," in *IEEE ISSCC Dig. Tech. Papers*, Feb. 2020, pp. 96–98.

[34] A. Margara, P. Peronio, G. Acconcia, G. Cugola, and I. Rech, "High-accuracy and video-rate lifetime extraction from time correlated single photon counting data on a graphical processing unit," *Rev. Sci. Instrum.*, vol. 90, no. 10, Oct. 2019, Art. no. 104709.

[35] H.-J. Choe, Y. Choi, D. J. Kwak, and J. Lee, "Prototype time-of-flight PET utilizing capacitive multiplexing readout method," *Nucl. Instrum. Methods Phys. Res. A, Accel. Spectrom. Detect. Assoc. Equip.*, vol. 921, pp. 43–49, Mar. 2019.

[36] L. Maggiani, C. Salvadori, M. Petracca, P. Pagano, and R. Saletti, "Reconfigurable architecture for computing histograms in real-time tailored to FPGA-based smart camera," in *Proc. IEEE 23rd Int. Symp. Ind. Electron. (ISIE)*, Jun. 2014, pp. 1042–1046.

[37] J. O. Cadenas, R. S. Sherratt, P. Huerta, and W.-C. Kao, "Parallel pipelined array architectures for real-time histogram computation in consumer devices," *IEEE Trans. Consum. Electron.*, vol. 57, no. 4, pp. 1460–1464, Nov. 2011.

[38] D. B. Younis and B. M. Younis, "Low cost histogram implementation for image processing using FPGA," *IOP Conf. Ser., Mater. Sci. Eng.*, vol. 745, no. 1, Feb. 2020, Art. no. 012044.

[39] J. Nickolls, I. Buck, M. Garland, and K. Skadron, "Scalable parallel programming with CUDA: Is CUDA the parallel programming model that application developers have been waiting for?" *Queue*, vol. 6, no. 2, pp. 40–53, Mar. 2008, doi: 10.1145/1365490.1365500.

[40] A. A. Khedkar and R. H. Khade, "High speed FPGA-based data acquisition system," *Microprocessors Microsyst.*, vol. 49, pp. 87–94, Mar. 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0141933116303453

[41] J. Lu, L. Zhao, K. Chen, P. Deng, B. Li, S. Liu, and Q. An, "Real-time FPGA-based digital signal processing and correction for a small animal PET," *IEEE Trans. Nucl. Sci.*, vol. 66, no. 7, pp. 1287–1295, Jul. 2019.

[42] A. C. Therrien, R. Herbst, O. Quijano, A. Gatton, and R. Coffee, "Machine learning at the edge for ultra high rate detectors," in *Proc. IEEE Nucl. Sci. Symp. Med. Imag. Conf. (NSS/MIC)*, Oct. 2019, pp. 1–4.

[43] M. Chandrashekar, U. Kumar, K. Reddy, and K. N. Raju, "FPGA implementation of high speed infrared image enhancement," *Int. J. Electron. Eng. Res.*, vol. 1, no. 3, pp. 279–285, 2002.

[44] J. Cadenas, S. Sherratt, P. Huerta, W.-C. Kao, and G. M. Megson, "C-slow retimed parallel histogram architectures for consumer imaging devices," *Trans. Consum. Electron.*, vol. 59, no. 2, pp. 291–295, May 2013.

[45] Q. Gan, J. M. P. Langlois, and Y. Savaria, "Parallel array histogram architecture for embedded implementations," *Electron. Lett.*, vol. 49, no. 2, pp. 99–101, Jan. 2013, doi: 10.1049/el.2012.2701.

[46] A. Shahbahrami, J. Hur, B. Juurlink, and S. Wong, "FPGA implementation of parallel histogram computation," in *Proc. 2nd HiPEAC Workshop Reconfigurable Comput.*, 2008, pp. 63–72.

[47] P. Mondal and S. Banerjee, "A reconfigurable memory-based fast VLSI architecture for computation of the histogram," *IEEE Trans. Consum. Electron.*, vol. 65, no. 2, pp. 128–133, May 2019.

[48] Xilinx. (2019). *7 Series FPGAs Memory Resources*. [Online]. Available: https://www.xilinx.com/support/documentation/user_guides/ug473_7Series_Memory_Resources.pdf

[49] Xilinx. (2018). *UltraScale Architecture Libraries Guide*. [Online]. Available: https://www.xilinx.com/support/documentation/sw_manuals/xilinx2018_1/ug974-vivado-ultrascale-libraries.pdf

[50] Xilinx. (2012). *Timing Closure User Guide*. [Online]. Available: https://www.xilinx.com/support/documentation/sw_manuals/xilinx14_7/ug612.pdf

[51] P. A. Simpson, *Timing Closure*. Cham, Switzerland: Springer, 2015, pp. 191–226, doi: 10.1007/978-3-319-17924-7_14.

[52] C. Ramamoorthy and H. F. Li, "Pipeline architecture," *ACM Comput. Surv.*, vol. 9, no. 1, pp. 61–102, Dec. 1977.

[53] Xilinx. (2020). *7 Series FPGAs Data Sheet: Overview*. [Online]. Available: https://www.xilinx.com/support/documentation/data_sheets/ds180_7Series_Overview.pdf

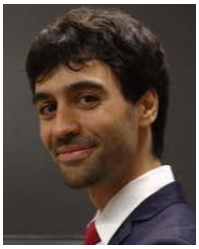
[54] F. Garzetti, N. Corna, N. Lusardi, and A. Geraci, "Time-to-digital converter IP-core for FPGA at state of the art," *IEEE Access*, vol. 9, pp. 85515–85528, 2021.

[55] N. Lusardi and A. Geraci, "Comparison of interpolation techniques for TDCs implementation in FPGA," in *Proc. IEEE Nucl. Sci. Symp. Med. Imag. Conf. (NSS/MIC)*, Oct. 2015, pp. 1–2.

[56] N. Lusardi, F. Garzetti, and A. Geraci, "The role of sub-interpolation for delay-line time-to-digital converters in FPGA devices," *Nucl. Instrum. Methods Phys. Res. A, Accel. Spectrom. Detect. Assoc. Equip.*, vol. 916, pp. 204–214, Feb. 2019.

[57] C. Hervé, J. Cerrai, and T. Le Caër, "High resolution time-to-digital converter (TDC) implemented in field programmable gate array (FPGA) with compensated process voltage and temperature (PVT) variations," *Nucl. Instrum. Methods Phys. Res. A, Accel., Spectrom., Detect., Assoc. Equip.*, vol. 682, pp. 16–25, Aug. 2012.

ANDREA COSTA (Member, IEEE) was born in Piacenza, in 1995. He received the B.Sc. degree in biomedical engineering and the M.Sc. degree in electronics engineering from Politecnico di Milano, in 2017 and 2020, respectively. His research interests include innovative hardware architectures for data processing in the field of FPGA time-domain devices and FPGA DAQ for high data rate environments.



NICOLA CORNA (Member, IEEE) was born in 1992. He received the bachelor's and master's degrees in electronics engineering from Politecnico di Milano, in 2015 and 2018, respectively, where he is currently pursuing the Ph.D. degree with DEIB, with a focus on the development of systems on FPGA and SoC re-configurable devices, particularly time-domain devices. He is the author and a developer of various open-source projects. His research interest includes free software.

FABIO GARZETTI (Member, IEEE) received the bachelor's and master's degrees in electronic engineering from Politecnico di Milano. He developed his thesis work at the Digital Electronics Laboratory of the DEIB on a topic regarding innovative solutions for calibration and triggering of asynchronous signals for time-to-digital converters (TDCs) in field programmable gate arrays (FPGA). In the same group, he applied for the award of a Temporary Research Fellowship within the framework of the "Design of modules for readout and processing of sampled data based on FPGA architectures" research programme, supported by CAEN ELS.

NICOLA LUSARDI (Member, IEEE) was born in Piacenza, in November 1990. He received the Ph.D. degree, in 2018. He developed his thesis work at the Digital Electronics Laboratory, DEIB, on a topic regarding high-resolution time-to-digital converters (TDCs) in field programmable gate arrays (FPGA).

He is a temporary Researcher with DEIB, a Professor of electronics with Politecnico di Milano, and an Associated Member with the Italian National Nuclear Physics Institute (INFN). He is a Co-Founder of TEDIEL S.r.l., an Italian start-up and spin-off of the Politecnico di Milano. His research line and knowledge as a digital designer have been acknowledged by public and private research centers. Since 2014, he has been collaborating with CERN in the LHCb Experiment, CAEN S.p.A., Viareggio, LU; CAELels S.r.l., Basovizza, TS; Elettra Sincrotrone Trieste S.C.p.A., Basovizza, TS; Single Quantum B.V., Delft, NL; the Technology University of Delft; École Polytechnique Fédérale de Lausanne; and Rete Ferroviaria Italiana.

ENRICO RONCONI (Member, IEEE) received the bachelor's and master's degrees in electronic engineering from Politecnico di Milano, in 2017 and 2020, respectively. His research interests include advanced programmable logic (PL) and software architectures for data processing and transfer in field programmable gate arrays (FPGA) implemented scientific equipment, currently used and developed together with the time-to-digital and digital-to-time converters (TDC and DTC) developed in the same laboratory.

ANGELO GERACI (Senior Member, IEEE) received the M.Sc. degree *(cum laude)* in electrical engineering and the Ph.D. degree *(cum laude)* in electronics and communication engineering from Politecnico di Milano, in 1993 and 1996, respectively. Since 2004, he has been an Associate Professor with the Department of Electronics, Information and Bioengineering (DEIB), Politecnico di Milano. His research interests include digital electronics based on microcontrollers, DSP and FPGA devices, specifically in the areas of radiation detection, medical imaging, energy storage for automotive electric systems, and HPC applications. He is a Lecturer of the ''sistemi elettronici digitali'' and ''digital electronic systems design'' courses with the School of Industrial and Information Engineering, Politecnico di Milano, and he holds courses for the PhD programme on Information Technology. He has been a scientific collaborator with the Italian National Nuclear Physics Institute (INFN), since 1995. He is a member of the Management Board and Deputy Coordinator of the Ph.D. School of Information Engineering, Politecnico di Milano. He is an Auditor of projects on behalf of the Italian MIUR. In 2003 he was elevated to the level of Senior Member of the IEEE Nuclear and Plasma Society. As a Referee for several international journals including IEEE TRANSACTIONS ON NUCLEAR SCIENCE and *Review of Scientific Instruments*, he is author and coauthor of more than 320 publications on refereed international congress proceedings and journals (IEEE Transactions Prize Paper Award by the IEEE Power Electronic Society, in 2004). He has been the advocate and manager of several sponsored joint research projects between the Politecnico di Milano and private/public companies.