Efficient random number generation techniques for CMOS single-photon avalanche diode array exploiting fast time tagging units

Andrea Stanco,[1,2] Davide G. Marangon,[1,*] Giuseppe Vallone,[1,2,3] Samuel Burri,[4] Edoardo Charbon,[4] and Paolo Villoresi[1,2,†]

[1]*Dipartimento di Ingegneria dell'Informazione, Università degli Studi di Padova, via Gradenigo 6B, 35131 Padova, Italy*
[2]*Istituto Nazionale di Fisica Nucleare (INFN) – sezione di Padova, Via Marzolo 8, 35131 Padova, Italy*
[3]*Dipartimento di Fisica e Astronomia, Università degli Studi di Padova, via Marzolo 8, 35131 Padova, Italy*
[4]*EPFL, Route Cantonale, 1015 Lausanne, Switzerland*

(Received 18 October 2019; accepted 3 April 2020; published 4 June 2020)

This work presents a technique to produce random bits by exploiting single-photon time of arrival. Two quantum random number generator (QRNG) devices based on the field programmable gate array (FPGA) technology are presented: Randy, which uses one discrete single-photon avalanche diode (SPAD), and LinoSPAD, which uses a complementary metal-oxide semiconductor (CMOS) SPAD array, along with a time-to-digital converter (TDC). Postprocessing procedures are explained in order to extract randomness, taking care of SPAD and TDC nonidealities. These procedures are based on the application of Peres [Ann. Statist. **20**, 590 (1992)] and Zhou and Bruck [arXiv:1209.0726] extraction algorithms. Achieved generation rates are 1.8 Mbit/s for the Randy device and 310 Mbit/s for the LinoSPAD device.

DOI: 10.1103/PhysRevResearch.2.023287

I. INTRODUCTION

In recent years, there has been widespread interest in random number generators based on physical processes of quantum nature. In fact, these devices, so-called quantum random number generators (QRNGs), represent the ultimate way to obtain reliable randomness, free from the typical nonrandom issues that affect algorithmic random number generators. Typically, QRNGs exploit the quantum properties of the optical radiation field in many "recipes." Different architectures exploit different properties; setups using entangled systems and violating Bell inequalities feature the highest unpredictability in the so-called device-independent (DI) framework [1–5]. Systems that trust the measurement apparatus but not the states of the quantum system or vice versa, the so-called semi-device-independent generators, work under less strict assumptions [6]: For this reason, they feature, in principle, less unpredictability than DI-QRNG but are more feasible to implement and reach even larger generation rates. The last category includes those generators that work in a framework of complete trust in both the quantum system and the measurement apparatus. This means that the absence of side information, exploitable by an adversary, is assumed. The most famous example of this kind of generator is the *welcher weg* (which path) QRNG that produces randomness according to which path a photon takes after interacting with a beam splitter [7,8]. In this work, we consider the subcategory of trusted QRNGs that were developed in order to limit the number of single-photon detectors. Random number generation with just one detector is indeed possible by exploiting time as an additional degree of freedom and by leveraging on the statistical features of the photon detection distribution. This could be done with the following procedure: time sampling of the photon detector with a sampling rate almost equal to the photon rate; application of dedicated generation protocols; and application of dedicated unbiasing algorithms. In this work, we will show that it is possible to generate true random numbers without the application of dedicated generation protocols. Using a higher sampling rate, we applied unbiasing algorithms directly to the stream of samples, achieving higher generation rate of true random numbers. We used this technique on two different systems: The first one, Randy, uses a standard clock to sample the signal of one single-photon detector; the second one, LinoSPAD, uses both a standard clock and a time-to-digital converter (TDC) to sample the signals from a matrix of 256 single-photon detectors.

The paper is structured as follows: In Sec. II, the most common ways to generate random numbers from time will be reviewed. In Sec. III, we will introduce our method and we will compare it with the previous ones. The results achieved by applying it to the numbers obtained with a single detector connected to an field programmable gate array (FPGA) will be presented. In Sec. V, we will extend this method to a matrix of 256 detectors with time tagging capabilities. In Sec. VI, the conclusions will be presented.

*Present address: Toshiba Europe Limited, Cambridge Research Laboratory, 208 Cambridge Science Park, Milton Road, Cambridge, CB4 0GZ, UK.
†paolo.villoresi@dei.unipd.it

II. EXISTING PARADIGMS OF SINGLE DETECTOR QRNGS

We now introduce two elaborated generation protocols, reported in literature, that can extract randomness from a photon distribution.

As a first example, let us consider the generation protocol introduced by Stipčević *et al.* [9], here referred to as "Diff-QRNG." The Diff-QRNG comprises a light source attenuated to single-photon level, illuminating a single-photon detector, and a clock that counts the time between the detection events. A binary random variable $X_j = \{0, 1\}$ is obtained by comparing the length of time intervals between three consecutive detections. It is therefore convenient to define the discrete random variable $\mathcal{T}^j$, associated to the detection instants, such that $\mathcal{T}^j = \{t_0^j, t_1^j, t_2^j\}$. Hence, X_j takes its value x_j according to the following "rule": Given $\Delta T_1 = t_1 - t_0$ and $\Delta T_2 = t_2 - t_1$,

(1) if $\Delta T_1 > \Delta T_2$ then $x_j = 0$,
(2) if $\Delta T_1 < \Delta T_2$ then $x_j = 1$,
(3) if $\Delta T_1 = \Delta T_2$ then $x_j = \emptyset$, i.e., no bit is generated.

The rule is iterated so that for the next bit b_{j+1} the new time interval starts with the end of the previous one, i.e., $t_2^j = t_0^{j+1}$. The physical principle that guarantees the identical ($\Pr[x_j = 0] = \Pr[x_j = 1]$) and independent ($\Pr[x_j|x_{j-1}] = \Pr[x_j|x_{j-1}, x_{j-2}, \ldots, x_1] = 1/2$) distribution of the bits follows from the memoryless property that characterizes the exponential distribution of the interarrival times ΔT, namely $\Pr[\Delta T_1 > \Delta T_2] = \Pr[\Delta T_2 > \Delta T_1] = 1/2$. As a consequence, a random string $X = \{X_1, X_2, \ldots, X_n\}$ with $n \to \infty$, is characterized by full Shannon entropy, i.e., $H(X) = n$ bits. This implies that the average number of independent and identically distributed (i.i.d.) bits that are generated per unit time is equal to $r_{\text{i.i.d.}} = r_{\text{phot}}/2$, where r_{phot} is the number of photodetections per unit time [10].

As a second example, let us consider the generation protocol introduced by Fürst *et al.* [11], here referred to as "OdEven-QRNG." As in the previous example, in the OdEven-QRNG a light source attenuated to a single-photon level illuminates a photomultiplier but in this case a counter enumerates the number of photons detected within a fixed time interval, τ, corresponding to the period of a sampling signal. Defining n_τ^j as the number of detections within the interval τ_j, a random binary variable X_j assumes its value x_j, with the following rule:

(1) if $n_\tau^j \mod 2 = 0$ then $x_j = 0$,
(2) if $n_\tau^j \mod 2 = 1$ then $x_j = 1$.

In other words, the bit value is determined according whether an even or odd number of detections is registered in the time interval τ. For a Poisson distribution, we can write the probability of having an even or odd number of detections:

(1) $\Pr[2n] = \sum_{n=0}^{\infty} \frac{(\lambda\tau)^{2n}}{(2n)!} e^{-\lambda\tau} = \frac{1+e^{-2\lambda\tau}}{2} = \Pr[x_j = 0]$,
(2) $\Pr[2n+1] = \sum_{n=0}^{\infty} \frac{(\lambda\tau)^{2n+1}}{(2n+1)!} e^{-\lambda\tau} = \frac{1-e^{-2\lambda\tau}}{2} = \Pr[x_j = 1]$,

where λ is the mean number of photons per second and $\lambda\tau = \langle n_\tau \rangle$ is the mean number of photons per time interval τ. To avoid a bias in the output, $\langle n_\tau \rangle$ has to be sufficiently large so that $\Pr[x_j = 0] \simeq \Pr[x_j = 1]$. Therefore, since $H(X)$ is a function of $\langle n_\tau \rangle$, the generation rate is given by $R = H(X)/(n\tau)$.

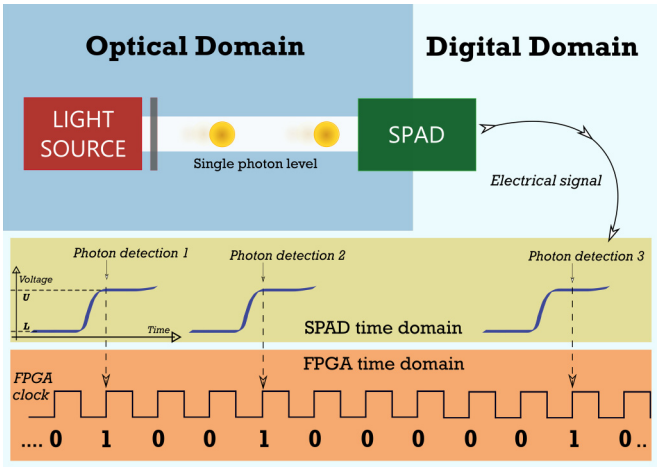

FIG. 1. A schematic view of Randy system. The SPAD links the optical domain to the digital domain by detecting photons from a light source attenuated to single-photon level and by sending electrical signals to the FPGA. Through its internal clock, the FPGA samples the SPAD events by its own time domain. Clock and SPAD rates are not on scale.

III. OUR APPROACH

Our contribution takes into consideration the simplest and most efficient way to generate random numbers by using the temporal degree of freedom without the need to devise complex rules. In its essence, the process of random number generation with a single-photon detector, for instance, a single-photon avalanche diode (SPAD), can be considered as a process with two signals: a squared-wave signal $\mathcal{S}_{\text{det}}(t)$ generated by the single-photon detector that is sampled on the rising edge of a square periodic signal $\mathcal{S}_{\text{clk}}(t)$, which is generated by a clock with a period τ_{clk}, as shown in Fig. 1. Without impinging photons $\mathcal{S}_{\text{det}}(t)$ has a typical value L. Given a photodetection at the time instant t_i, the $\mathcal{S}_{\text{det}}(t)$ toggles its state from $\mathcal{S}_{\text{det}}(t)(t < t_i) = L$ to $\mathcal{S}_{\text{det}}(t)(t_i \leqslant t < \tau_U) = U$. τ_U is the specific fixed time interval in which a SPAD keeps the state U before returning to L and typically $\tau_U < \tau_{\text{dead}}$, being τ_{dead} the SPAD dead time. This means that $\mathcal{S}_{\text{det}}(t)$ toggles back to the L state before the SPAD is able to detect another photon. The binary random variable X_j takes its value x_j at the instant $j\tau_{\text{clk}}$ according to the following rule:

$$x_j = \begin{cases} 1 & \text{if } \mathcal{S}_{\text{det}}(j\,\tau_{\text{clk}}) = U \neq \mathcal{S}_{\text{det}}((j-1)\,\tau_{\text{clk}}) \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

with the index $j \in \{0, 1, 2, \ldots\}$. The above rule is equivalent to having $x_j = 1$ whenever the clock detects a rising edge of $\mathcal{S}_{\text{det}}(t)$ and $x_j = 0$ otherwise. It is worth noticing that all existing paradigms of QRNG based on photon time of arrival can be seen as a (nonoptimal) postprocessing algorithm of the sequence $X = (x_1, x_2, \ldots, x_n, \ldots)$. We note that the maximum content of randomness that can be extracted by the above physical process is fully included in the X sequence, and in particular it is given by the Shannon entropy (in the large n limits):

$$\frac{H(X)}{n} = -[p_0 \log_2(p_0) + p_1 \log_2(p_1)], \tag{2}$$

where p_0 and p_1 are the probability of obtaining $x_j = 0$ and $x_j = 1$ respectively.

It is clear that the maximum generation rate is given by $r_{\max} = \tau_{\text{clk}}^{-1}$ (in case of a perfect detector). Given a mean photon number per second $\lambda = r_{\text{phot}}$, for a Poisson distribution the probability of no detections within the τ_{clk} interval is equal to $\Pr[\emptyset] = e^{-r_{\text{phot}}\tau_{\text{clk}}}$. To achieve the rate $r_{\max}$, it is necessary to avoid a bias in the output string X by tuning the photon detection rate; this is to obtain at least one detection within a sampling period, namely $1 - \Pr[\emptyset] = 1/2$ from which we obtain $r_{\text{phot}} = \ln 2/\tau_{\text{clk}}$. It is therefore clear that as the clock rate increases, the detection rate should become larger to keep the bias low in 0. However, r_{phot} cannot be set arbitrarily large as it is strongly limited by the detector dead time. Given a fixed r_{phot}, our idea is to increase the generation rate r by deliberately increasing the sampling rate and producing a highly biased string with plenty of 0's and very few 1's. Then, we rebalance the bias through an optimal postprocessing algorithm, introduced by Peres [12]. Peres' algorithm is a revision of the famous von Neumann algorithm [13] and allows the removal of bias from a string while extracting the maximum entropy from that. We give a brief description of it. Given a sequence of samples s_w, Peres procedure is defined as

$$\Psi(s_w) = \Psi_N(s_w)||\Psi(\Psi_U(s_w))||\Psi(\Psi_V(s_w)) \tag{3}$$

where $\Psi_N(s)$ is the von Neumann algorithm applied to the string, $\Psi(\Psi_U(s))$ is the Peres algorithm applied to a substring $\Psi_U(s)$, and $\Psi(\Psi_V(s))$ is the Peres algorithm applied to another substring $\Psi_V(s)$. $\Psi_U(s)$ is created by an exclusive OR (XOR) operation on every bit-pair of the string, while $\Psi_V(s)$ is created by discarding one bit of each 00 and 11 pair. For a full description of the procedure, refer to Ref. [12]. Clearly, the i.i.d. property comes from the Poisson distribution itself.

The extraction rate after Peres algorithm, under asymptotic assumptions, is given by

$$r_{\text{i.i.d.}} = \tau_{\text{clk}}^{-1}\frac{H(X)}{n}. \tag{4}$$

As shown in Fig. 2, by fixing r_{phot}, the extraction rate increases with the sampling rate despite the value of $H(X)$ (the maximum entropy is reached with a sampling rate equal to $r_{\text{phot}}/\ln 2 = 288.539\,\text{kHz}$). These plots confirm the following: With a fixed r_{phot}, the generation rate is more influenced by the actual input length than by the bias value. Therefore, the optimal choice is to have $\tau_{\text{clk}}^{-1} \gg r_{\text{phot}}$ with no upper bound since $r_{\text{i.i.d.}} \to \infty$ when $\tau_{\text{clk}} \to 0$. However, increasing the sampling frequency implies a logarithmic improvement of the generation rate (see Fig. 2) and using frequencies that are too large could be ineffective.

To summarize, any QRNG based on the photon time of arrival can be modeled by a binary signal $\mathcal{S}_{\text{det}}(t)$ sampled by a clock $\mathcal{S}_{\text{clk}}(t)$ that gives an output sequence X. Standard generation protocols, such as Diff-QRNG or OdEven-QRNG, can be applied to X. Nevertheless, these protocols are far from being efficient. Here we propose an optimal postprocessing able to extract the maximum available entropy from the string X based on the Peres algorithm. In the next section, we will show how to apply our method to physical generators, while dealing with the nonidealities of the detectors.

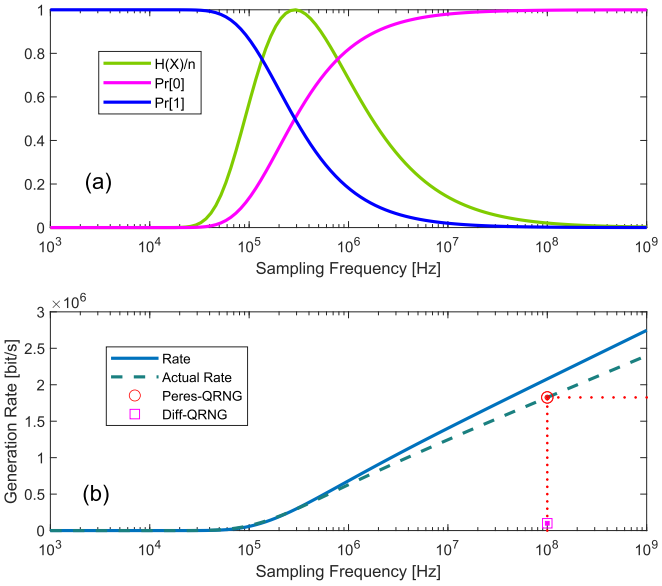

FIG. 2. The upper plot shows the balance between 0's and 1's (Pr[∅] and Pr[1]) as well as the binary entropy as function of the sampling rate. The photon count rate is fixed to 200 kcounts/s. In the lower plot, the continuous line shows the theoretical generation rate as function of the sampling rate before any afterpulses or dead time treatment. The dashed line shows the actual generation after the afterpulses and dead time removal. The expected 1.815 Mbit/s rate after Peres postprocessing is highlighted by the dotted line. The magenta square highlights the rate obtained by the Diff-QRNG protocol.

IV. SINGLE DETECTOR IMPLEMENTATION

We first implemented our method by using a single SPAD illuminated by an attenuated laser light. We designed a system that is capable of producing true random numbers using different generation protocols as well as the Peres algorithm. The system was developed from an FPGA/CPU device [14]. The FPGA allows us a full control over the generation process and great flexibility in order to easily switch from one protocol to another. We called this design "Randy." A schematic view of the setup is shown in Fig. 1. We set the light source intensity to keep the photon count rate around 200 kcounts/s, due to the SPAD nonlinear behavior on higher rates. As a first implementation, we used the default 100-MHz system clock to sample the SPAD signals. The advantages are quite clear since an FPGA allows a full description of the time evolution of the system: Every operation can be described as a multiple of the fundamental time unit τ_{clk} defined by the system clock. Therefore, the behavior of the system is fully deterministic apart from the nondeterministic side due to the true randomness of the photon time of arrival.

The FPGA saves a logical 0 for every clock cycle whenever the SPAD signal is low, and a logical 1 otherwise. The raw string is then postprocessed on a dedicated CPU. Because of the huge difference between the clock rate and the photon rate (100 MHz over 200 kcount/s), the raw strings are heavily biased in zero with a percentage of 99.8%. Ideally, the role of Peres algorithm is to reduce the bias to zero. Nevertheless, the Peres algorithm cannot work around any correlation. On

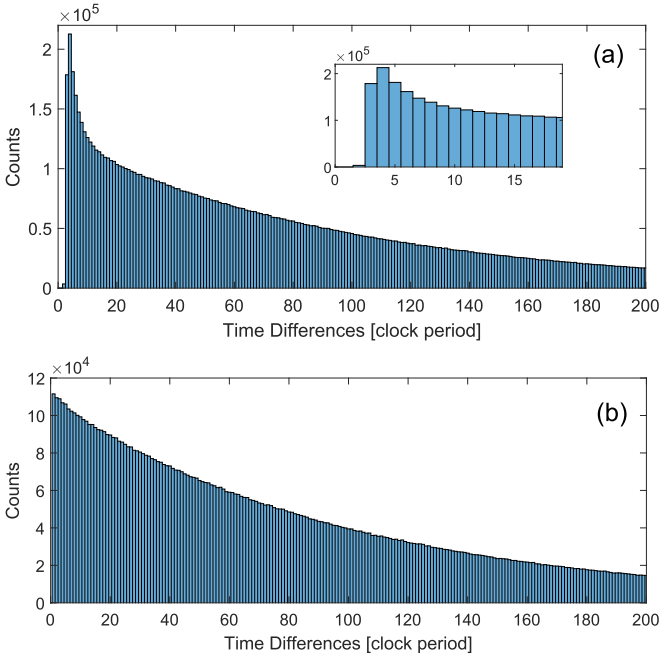

FIG. 3. Top plot: time differences histogram before any post-processing. The dead time causes the gap at zero with no events occurring within three clock cycles. It is also clear the presence of afterpulses which appear as a deviation of the histogram from an ideal exponential. Indeed, the curve trend shows a higher count rate below the 18 clock cycles threshold. Bottom plot: time difference histogram after the removal of afterpulses and dead time.

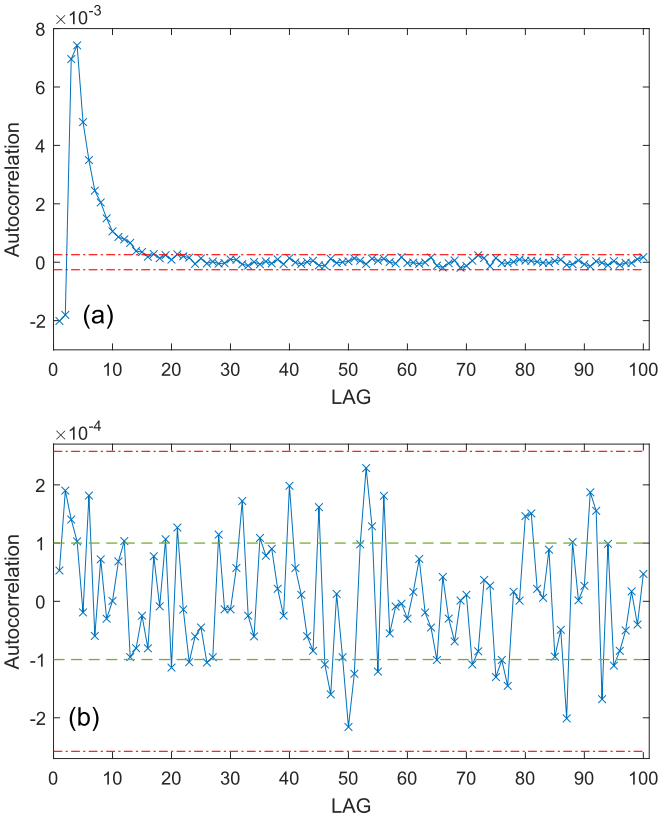

FIG. 4. Top plot: the serial correlation evaluated on a sampled bit string without any dead time or afterpulse treatment. Bottom plot: the serial correlation evaluated on the same sampled bit string after dead time and afterpulse removal. The serial correlation re-enters within the limit of acceptance. Green dashed lines represent standard deviation while red dot-dashed lines are 99% confidential limits.

the contrary, it could emphasize it. Therefore, any correlation has to be removed before the application of Peres algorithm. In our case, correlation comes from afterpulses and dead time of the detectors, and we describe how to remove them in the following.

We evaluated the distribution of the time interval between consecutive events. For ideal Poissonian events, this distribution is expected to be a decreasing exponential. As shown in Fig. 3, the experimental distribution differs from the ideal distribution by two effects.

The first one is the *dead time*, whose value depends on the specific SPAD used [15], and represents the minimum time distance between two rising edge of $\mathcal{S}_{\rm det}(t)$. $\tau_{\rm dead} \simeq 30$ ns, corresponding to three clock cycles.

The second effect is the *afterpulse*. Afterpulses are spurious events that occur randomly within a fixed time interval from a real detection. As a result, they produce a peak which undermines the exponential trend of the events' difference distributions. We evaluated an 18 clock cycles (180 ns) cross-point between the only-true-events region and the afterpulses region.

Dead time and afterpulses introduce correlations in the output string. Clearly, after a value $x_j = 1$, some of the few following bits in X are not independent. Since we estimated 18 clock cycles as the required time to be in the true-event region, we removed 18 values of X following any value $x_j = 1$ to get rid of the correlation.

As a result, this procedure eliminated a portion of valid random events which we evaluated to be around 14%. However, the resulting distribution of the difference between consecutive 1's follows the expected decreasing exponential (see Fig. 3 bottom). Moreover, the process eliminates the correlation initially present in the string X, as Fig. 4 demonstrates.

After compensating dead time and afterpulse, we applied Peres algorithm to the bit string. With an actual photon count rate of 172 kcounts/s and an equivalent sampling frequency of 97 MHz (due to afterpulses and dead time removal), a real-time implementation of Peres algorithm would have yielded a final true random bit rate of 1.8 Mbit/s according to the rates shown in Fig. 2. We point out that the realization of a real-time version of this procedure requires us to address some challenges. Indeed, in order to have a valuable generation efficiency (>99%), it is required to store and process sequences larger than 10^8 bit. Therefore, both memory and computational complexity have to be addressed. Furthermore, the evaluation of Peres computational costs is not trivial and may change according to different input sequences [16]. As shown in Fig. 4, the correlation over the output string is within the limits of statistical acceptance. The resulting unbalance between 0's and 1's is less than 0.01% likely.

Moreover, the two generation protocols Diff-QRNG and OdEven-QRNG described in Refs. [9,11] were also implemented on the same FPGA system in order to compare

their performances with our method. The high time resolution ($r_{\text{phot}} \ll \tau_{\text{clk}}^{-1}$) allows enough precision to identify time differences and to have a good estimation on the detected photons within a fixed intervals. Indeed, the system was successfully used to produce timed random numbers in the work of Vedovato *et al.* [17]. Of course, the only uncertainty came from the photons' time of arrival, but it was handled by setting the photon counting rate to a proper value [18]. On the other hand, given the same optical setup and clock frequency and assuming an ideal detector (i.e., no dead time or afterpulses), these protocols have lower bitrate performances: a photon count rate of 200 kcount/s produces a rate of true random bits of 100 kbit/s for the Diff-QRNG protocol and of 20 kbit/s for the OdEven-QRNG [19].

V. MULTIPLEXING THE GENERATION RATE

The main limitation concerning the use of QRNG based on SPADs, i.e., discrete variable QRNG, is the limited generation rate achievable. Typically, SPADs feature maximum count rates of a few Mcps (counts per second). QRNGs based on continuous variable (CV) protocols are therefore preferred when it comes to obtain rates in the order of Gbps [20,21]. However, the recent advancements in miniaturization techniques, and especially the creation of deep-submicron complementary metal-oxide semiconductor (CMOS) SPADs, have led to arrays and matrices with hundreds or even thousands of SPADs [22,23]. Hence, each SPAD can be considered as a pixel of an extremely sensitive light sensor. The application to the case of random number generation is then straightforward: Given that every pixel works independently, it is possible to multiplex the random signals and then fill the rate gap with CV-QRNGs [24,25]. The typical approach is to generalize the paradigm of the *welcher weg* QRNG—a beam splitter and two photon paths—to a generator where photons can take N possible paths, with N being the total pixel number of the sensor [26]. Nevertheless, as in Randy, the temporal degree of freedom can be exploited as well, allowing an easier calibration. Therefore, the sensor is illuminated with a uniform light intensity so that each SPAD has the same probability to click within a given time interval, which corresponds to the exposure time for a frame. Random numbers are indeed produced by periodically sampling each pixel and applying dedicated generation protocols.

In this work, we consider a sensor of recent introduction, LinoSPAD [27], from the AQUA laboratory at Delft University and EPFL, which features a linear array of 256 pixels connected to a single FPGA. The peculiarity of LinoSPAD is a time tagging functionality, which associates a temporal coordinate to every detection, thus potentially enabling a further increase in the generation rate. In the following, we will apply the techniques described in the previous sections for the single-detector case to LinoSPAD.

LinoSPAD features 64 FPGA-based time-to-digital converters [28,29] that tag the detections of each SPAD in a given bank. Each TDC is implemented by a delay line with 35 carry elements of 4 bits and it is sampled with a frequency $f_{\text{clock}} =$ 400 MHz. At every time interval $\tau_{\text{clock}} = 2.5$ ns the TDC emits an output code $b \in [0, 139]$. The TDC has therefore a sub-

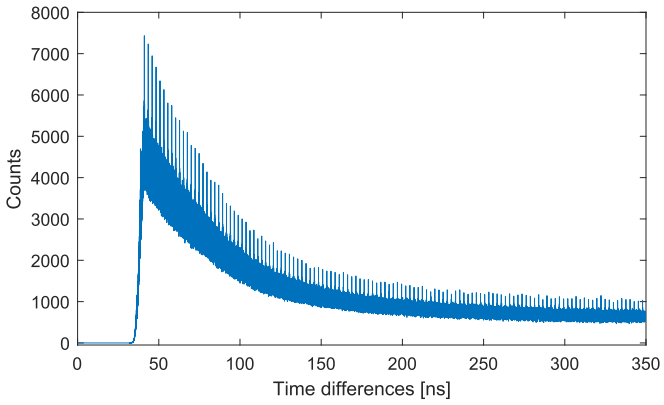

FIG. 5. Distribution of the time differences between successive detections on LinoSPAD device.

resolution of $\tau_{\text{sub}} = \tau_{\text{clock}}/140 \simeq 17.86$ ps and this represents the fundamental time resolution of the system. The following considerations take into account that the actual number of valid pixels is 64 and not 256 due to the limitation of 64 TDCs. The measurements of the photon time of arrival are taken with respect to a "reference" time signal, whose period determines the integration time of a frame. The buffer of the system can add output codes to a maximum of 2^{28} bins. Hence, the longest measurable time interval between the reference clock signal and a photon detection cannot be larger than $\tau_{\text{sub}} 2^{28} \simeq 4.8$ ms.

Since the device registers a maximum of 512 tags per pixel during the integration time, every frame is composed at most by 64×512 tags. Similar to the paradigm adopted in the previous sections, random bits can be extracted directly from the bare physics of the process: A string $\tau_{\text{frame}}/\tau_{\text{sub}}$ bits long is associated to each frame and the 1's, equal in number to the number of tags, are located according to the tag values. Again, the limit of this approach is that the strings are consistently biased toward zero. This bias is the result of two concurring causes: The first one is the dead time of the SPADs, which being of $\tau_{\text{dead}} \simeq 40$ ns, implies that every bit 1 is necessarily followed by $\tau_{\text{dead}}/\tau_{\text{sub}} \simeq 2240\,0'$s. However, the 0's due to the dead time can be removed, as was previously done. The second cause is the limited buffer size: Each string produced in a frame will always feature at most 512 1's. Indeed, an extraction approach by means of Peres debiasing procedure could be suitable even for this framework. As in Randy, we studied the interarrival detections time, whose distribution is reported in Fig. 5, in order to detect the artifacts induced by the physical limits of the device.

The histogram starts approximately at 40 ns due to the dead time, and a peak starting at 40 ns and extending up to 200 ns indicates the presence of afterpulses. A noticeable feature is an unusual peak pattern. This pattern was due to a nonlinear behavior of the TDC (further details in Sec. V B). Hence, to manage these nonidealities we decided to separate the tag resolution between *coarse* resolution (clock sampling) and *fine* resolution (TDC), implementing two different postprocessing procedures. This distinction allows an easy postprocessing procedure since it separates the treatment of nonidealities, i.e., *coarse* for detectors ones and *fine* for electronics ones (see Secs. V A and V B). The analysis was done on data collected

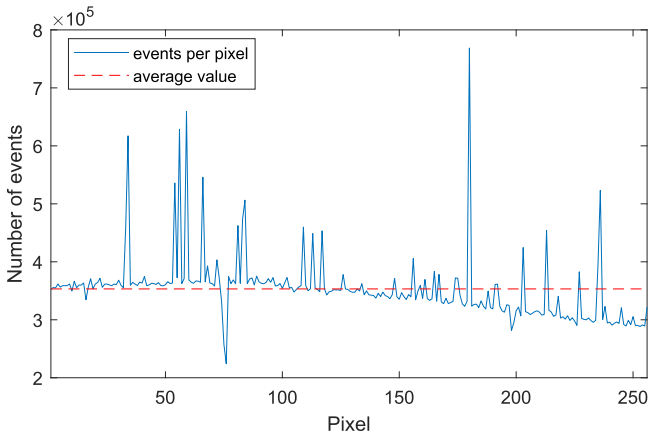

FIG. 6. Number of events for every pixel. From the graph, it is clear that the efficiency varies from pixel to pixel with highly efficient isolated pixels and a descending trend.

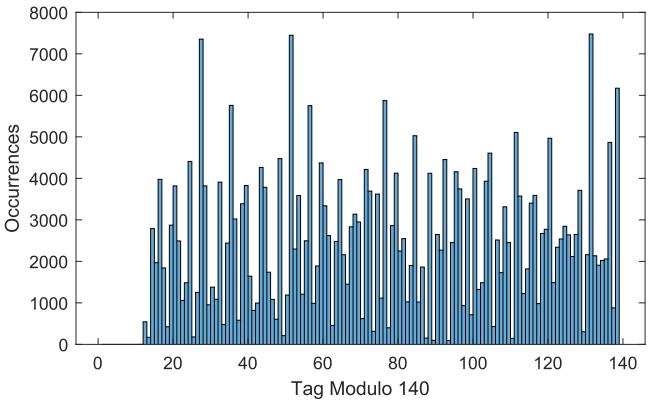

FIG. 7. Distribution of the tags modulo 140 of the pixel 256. The highly biased distribution shows the nonlinear response of the TDC.

with an acquisition of 8×10^3 frames where every frame has an integration time $\tau_{\text{frame}} = 320 \times 10^{-6}$ s. The photon rate was tuned to obtain approximately 400 counts per frame. Given a buffer size of 512 detections, this value was chosen in order to keep low the probability of saturation and frame loosing. Results show a final achieved generation bit rate equal to 310 Mbit/s.

A. Coarse resolution

The *coarse* resolution is defined by the 400-MHz system clock. The tag information is related to the number of clock cycles in which an event is detected. Therefore, it describes the temporal distance between an event and the zero reference with a resolution of 2.5 ns. We remove dead time and afterpulses of the SPADs with the same technique described in Sec. III. However, the SPAD arrays suffer from *pixel cross correlation,* which causes a pixel to output an event when its neighbor receives a photon, i.e., it produces a fake event as in the afterpulse. In order to remove it and to be sure that no cross correlation exists, we discard approximately one third of the pixels of a bank, losing all the events from those pixels. This procedure reduces the actual number of pixel from 64 to 22. After removing the cross correlation, the dead time, and the afterpulse, we applied the Peres algorithm to every selected pixel independently. As discussed in Sec. III and according to Fig. 2, higher rates can be achieved by preferring longer heavily biased bit strings over shorter slightly biased bit strings. Therefore, we treat every selected pixel as an autonomous QRNG. As in single-detector implementation, we evaluated the serial correlation on a sampled bit string and it re-enters within the limit of acceptance after removing dead time, afterpulse, and pixel cross correlation. By summing the bit rate of the selected pixels, we obtain a total bit rate of $\mathcal{R}_{\text{coarse}} \simeq 87$ Mbit/s. The value is averaged since the event rate varies from pixel to pixel, as shown in Fig. 6. The mean extraction rate per pixel is equal to $\mathcal{R}_{\text{coarse}}/64 = \mathcal{R}_{\text{coarse},p} \simeq 1.36$ Mbit/s. Considering an ideal SPAD array with no correlation, the hypothetical extraction rate per pixel would be equal to $\mathcal{R}_{\text{coarse}}/22 = \mathcal{R}^*_{\text{coarse},p} \simeq 3.95$ Mbit/s.

B. Fine resolution

The *fine* resolution is defined by the TDC and has a time resolution of approximately 17.86 ps. The TDC outputs a number between 0 and 139, which identifies a precise moment within a clock cycle in which an event occurred. Hence, the access to the TDC value is done at every clock cycle. As in the coarse resolution, we decide to treat every pixel independently. The peaks of Fig. 5 are exactly separated by 2.5 ns, which can be explained by the presence of a dead time in accessing the TDC. This behavior brings to the tag distribution shown in Fig. 7, which represents an entire 2.5-ns clock cycle. The figure shows a significant bias on several values and no events in the lower bins. This behavior is due to the TDC implementation on an FPGA technology which introduces nonlinearity caused by different propagation delays over hardware blocks [30]. It is worth noticing that, while the distribution is not uniform, its entropy is equal to $H_{\text{exp}} = -\sum_{k=0}^{139} p_k \log_2 p_k \simeq 6.8$ bits, very close to the maximum entropy $H_{\text{th}} = \log_2 140 \simeq 7.12$ bits achievable with 140 uniformly distributed decimal values. We also point out that, if such 140 integers are transformed into their binary description, the biased distribution introduces a correlation of the bits. Numbers in decimal basis are just biased and not correlated; by transforming the integers into a binary basis, the resulting bits are correlated due to the nonuniform original distribution. Moreover, in order to describe 140 different values, 8 bits are required, implying that there are no events from 140 to 255, which worsens the situation. Clearly, the Peres algorithm cannot be applied to this string.

In order to remove the bias and the correlation on the modulo 140 distribution, it is possible to use another post-processing method, the algorithm proposed by Zhou and Bruck [31]. The latter is the generalization of the Peres algorithm for biased distributions over a finite number of integers. Now we briefly describe the Zhou-Bruck algorithm (for a full description of the method, refer to Ref. [31]). Let us consider a random variable X with n possible outcomes with a biased probability distribution. Let us define $b = \lceil \log_2 n \rceil$ as the number of bits required for a binary description of the outcomes (in our case with $n = 140$ we have $b = 8$). If the outcomes are labeled as 0, 1, ..., $n-1$ and converted to binary, a string of b bits corresponds to each outcome.

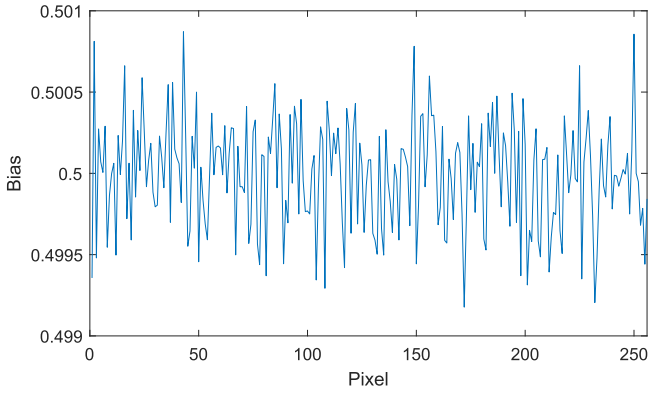

FIG. 8. Bias after the application of the Zhou-Bruck method. The graph shows the perfect balancing between 0's and 1's for every pixel within the order of 10^{-4}.

For a given sequence $(x_1, \ldots, x_M)$, we can convert any x_k to the corresponding binary string and consider only the first bit of each string: The resulting sequence is biased but has no correlation. Let us now consider the second bit of each string. We may create two sequences corresponding to the two possible values of the first bit: The first sequence collects the second bits related to a 0 first bit and vice versa for the second one. Again, these two sequences have bias but no correlation. The idea can be iterated: The ith bits can be grouped into $2^i - 1$ sequences according to the values of the $i - 1$ bits. After this manipulation, one gets N different sequences where $N = 2^b$. Since these N sequences have bias but no correlation, the Peres algorithm can be applied separately to each of them (if the sequence is not empty). We implemented the Zhou-Bruck method on the tags modulo 140 of each pixel. As a matter of fact, Zhou-Bruck method is quite efficient. For example, for the pixel 256, from each tag we get an average of 6.3 unbiased bits; this value is close to the maximum $H_{\text{exp}} \simeq 6.8$ bits that can be extracted with a perfect efficient algorithm. The obtained bits were evaluated in term of bias (Fig. 8) and binary entropy extraction efficiency (Fig. 9) as well as serial correlation, which re-enters within the limit of acceptance.

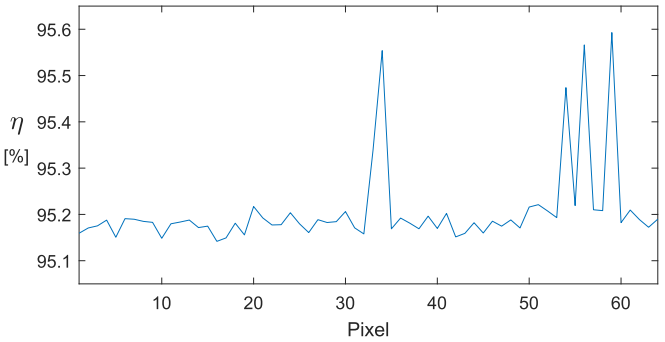

FIG. 9. Extraction efficiency of the Zhou-Bruck method for every pixel. The efficiency is evaluated as $\eta(p) = N_{\text{bit}}/(N_{\text{tag}} H_{\text{exp}})$, where N_{bits} and N_{tags} are the number of extracted bits and the number of tags. The four spikes are due to a higher count efficiency of the selected pixels according to Fig. 6.

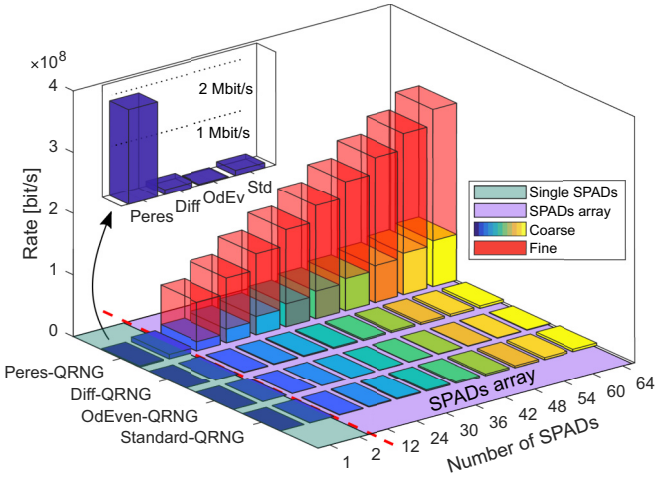

FIG. 10. Bar diagram comparing the rates of different protocols. The diagram shows the qualitative bit rates differences between different protocols. It is divided in two areas which represent the use of discrete SPADs (as in Randy) and SPAD array (LinoSPAD).

The final rate of the *fine* resolution is equal to $\mathcal{R}_{\text{fine}} = 223\,\text{Mbit/s}$ for the whole pixels array. In order to evaluate the rate for a single pixel, it must be taken into account that there are only 64 TDCs which switch among the four pixel banks. Therefore, the actual number of pixels (in terms of rate) truly is 64 and the final rate per pixel is $\mathcal{R}_{\text{fine}}/64 = \mathcal{R}_{\text{fine},p} \simeq 3.48\,\text{Mbit/s}$ per pixel. Thus, by considering the *fine* and *coarse* resolution, the average generation rate per pixel is given by $\mathcal{R}_{\text{tot},p} = \mathcal{R}_{\text{fine},p} + \mathcal{R}_{\text{coarse},p} \simeq 4.84\,\text{Mbit/s}$ while the total generation rate for LinoSPAD device is given by $\mathcal{R}_{\text{tot}} = \mathcal{R}_{\text{fine}} + \mathcal{R}_{\text{coarse}} \simeq 310\,\text{Mbit/s}$.

VI. CONCLUSION

In this paper, we showed improved techniques to produce true random numbers by processing single-photon events. An innovative CMOS SPAD array device called LinoSPAD was used to implement a high-rate RNG. It integrates a temporal tagging system with a detector matrix, which allows the usage of the temporal degree of freedom in addition to the much more common spatial one. Starting from a single detector system (Randy), we defined an efficient procedure to fully exploit the temporal degree of freedom. Compared to existing paradigms which are based on complex rules and are far from being information efficient, our procedure extracts the most of the system entropy, achieving the maximum bit rate allowed by the system. This procedure is based on the use of a high-frequency sampling clock (compared to the photon rate) and on the use of the Peres unbiasing algorithm. Therefore, the generation rate is only limited by the physical device performances and not by the technique itself. Applying this technique to LinoSPAD required a further step to deal with detectors matrix nonidealities (pixel cross correlation) and TDC ones. Hence, a dedicated postprocessing procedure, which included the usage of the Zhou-Bruck algorithm, was developed in order to work around such nonidealities. The summary comparison bar diagram of Fig. 10 clearly

shows the remarkable differences among different generation procedures. Our Peres-based procedure clearly reaches a higher bit rate compared to a protocol-based one. Furthermore, moving from the discrete SPADs framework (Randy) to the CMOS SPAD array one and increasing the sampling frequency as well as adding a time-to-digital converter (LinoSPAD) improve the generation rate even more. Final results show a bitrate per SPAD/pixel equal to the following:

(1) $\mathcal{R}_{\text{Randy,spad}}(100\,\text{MHz}) = 1.8\,\text{Mbit/s}$,

(2) $\mathcal{R}_{\text{LinoSPAD,spad}}(400\,\text{MHz} + \text{TDC}) = 4.84\,\text{Mbit/s}$,

and for LinoSPAD a total bit rate of $\mathcal{R}_{\text{LinoSPAD}} = \mathcal{R}_{\text{LinoSPAD,spad}} 64 = 310\,\text{Mbit/s}$.

Moreover, applying these techniques to other physical devices with better performances will further increase the generation rate. Future steps will also consider a real-time implementation of the two procedures since the preprocessing and both the Peres and Zhou-Bruck algorithms could be effectively implemented via FPGA.

ACKNOWLEDGMENT

Part of this work was supported by Ministero dell'Istruzione, dell'Università e della Ricerca (MIUR) (Italian Ministry of Education, University and Research) under the initiative "Departments of Excellence" (Law 232/2016).

[1] S. Pironio, A. Acín, S. Massar, A. B. de la Giroday, D. N. Matsukevich, P. Maunz, S. Olmschenk, D. Hayes, L. Luo, T. A. Manning, and C. Monroe, Nature (London) **464**, 1021 (2010).

[2] B. G. Christensen, K. T. McCusker, J. B. Altepeter, B. Calkins, T. Gerrits, A. E. Lita, A. Miller, L. K. Shalm, Y. Zhang, S. W. Nam, N. Brunner, C. C. W. Lim, N. Gisin, and P. G. Kwiat, Phys. Rev. Lett. **111**, 130406 (2013).

[3] P. Bierhorst, E. Knill, S. Glancy, Y. Zhang, A. Mink, S. Jordan, A. Rommal, Y.-K. Liu, B. Christensen, S. W. Nam, M. J. Stevens, and L. K. Shalm, Nature (London) **556**, 223 (2018).

[4] Y. Liu, X. Yuan, M.-H. Li, W. Zhang, Q. Zhao, J. Zhong, Y. Cao, Y.-H. Li, L.-K. Chen, H. Li, T. Peng, Y.-A. Chen, C.-Z. Peng, S.-C. Shi, Z. Wang, L. You, X. Ma, J. Fan, Q. Zhang, and J.-W. Pan, Phys. Rev. Lett. **120**, 010503 (2018).

[5] S. Gómez, A. Mattar, E. S. Gómez, D. Cavalcanti, O. J. Farías, A. Acín, and G. Lima, Phys. Rev. A **97**, 040102(R) (2018).

[6] X. Ma, X. Yuan, Z. Cao, B. Qi, and Z. Zhang, npj Quantum Inf. **2**, 16021 (2016).

[7] J. Rarity, P. Owens, and P. Tapster, J. Mod. Opt. **41**, 2435 (1994).

[8] T. Jennewein, U. Achleitner, G. Weihs, H. Weinfurter, and A. Zeilinger, Rev. Sci. Instrum. **71**, 1675 (2000).

[9] M. Stipčević and B. M. Rogina, Rev. Sci. Instrum. **78**, 045104 (2007).

[10] With the exception of bit x_1 and x_n, two photodetections are necessary to generate an i.i.d. bit.

[11] H. Fürst, H. Weier, S. Nauerth, D. G. Marangon, C. Kurtsiefer, and H. Weinfurter, Opt. Express **18**, 13029 (2010).

[12] Y. Peres, Ann. Statist. **20**, 590 (1992).

[13] J. von Neumann, in *Monte Carlo Method*, National Bureau of Standards Applied Mathematics Series Vol. 12, edited by A. S. Householder, G. E. Forsythe, and H. H. Germond (U.S. Government Printing Office, Washington, DC, 1951), Chap. 13, pp. 36–38.

[14] We used the ZedBoard produced by Avnet.

[15] We used the SPCM-ARQH by Excelitas.

[16] A. Prasitsupparote, N. Konno, and J. Shikata, Entropy **20**, 729 (2018).

[17] F. Vedovato, C. Agnesi, M. Schiavon, D. Dequal, L. Calderaro, M. Tomasin, D. G. Marangon, A. Stanco, V. Luceri, G. Bianco, G. Vallone, and P. Villoresi, Sci. Adv. **3**, e1701180 (2017).

[18] The time required by the FPGA in order to implement the protocols was totally negligible.

[19] The time interval was set in order to have $\langle n_\tau \rangle \simeq 10$ according to what was stated in Sec. II.

[20] D. G. Marangon, G. Vallone, and P. Villoresi, Phys. Rev. Lett. **118**, 060503 (2017).

[21] M. Avesani, D. G. Marangon, G. Vallone, and P. Villoresi, Nat. Commun. **9**, 5365 (2018).

[22] C. Niclass, M. Sergio, and E. Charbon, in *Proceedings of the Design Automation & Test in Europe Conference* (IEEE, Munich, Germany, 2006), pp. 1–6.

[23] E. Charbon, Philos. Trans. R. Soc. A **372**, 20130100 (2014).

[24] D. Stucki, S. Burri, E. Charbon, C. Chunnilall, A. Meneghetti, and F. Regazzoni, Proc. SPIE **8899**, 88990R (2013).

[25] S. Burri, D. Stucky, Y. Maruyama, C. Bruschini, E. Charbon, and F. Regazzoni, in *International Image Sensor Workshop, Utah, USA, 12-16 June*, EPFL-CONF-191217 (EPFL scientific publications, 2013).

[26] D. G. Marangon, G. Vallone, U. Zanforlin, and P. Villoresi, Quantum Sci. Technol. **1**, 015005 (2016).

[27] S. Burri, H. Homulle, C. Bruschini, and E. Charbon, Proc. SPIE **9899**, 98990D (2016).

[28] C. Favi and E. Charbon, in *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays, FPGA '09* (ACM, New York, 2009), pp. 113–120.

[29] M. Fishburn, L. H. Menninga, C. Favi, and E. Charbon, IEEE Trans. Nucl. Sci. **60**, 2203 (2013).

[30] J. Song, Q. An, and S. Liu, IEEE Trans. Nucl. Sci. **53**, 236 (2006).

[31] H. Zhou and J. Bruck, arXiv:1209.0726.