Real-Time Data-Acquisition Platform for Pulsed Measurements

Sergey V. Polyakov[a], Alan Migdall[a] and Sae Woo Nam[b]

[a]*Joint Quantum Institute, University of Maryland, College Park, MD 20742 and Optical Technology Division, NIST 100 Bureau Dr., Gaithersburg, MD 20899*
[b]*Optoelectronics Division, NIST 325 Broadway, Boulder, CO 80305*

Abstract. We present an inexpensive and simple data acquisition platform based on Field Programmable Gate Arrays (FPGAs) designed to acquire and characterize fast digital or analog electrical signals in real time for processing on a generic personal computer. While the instrument was designed for electrical outputs of single-photon detectors and is suited for high photon-counting rates, it can also be used for characterization of similar digital electrical signals from other sources and for analog signals as well. The complete description of the platform is available for download at http://physics.nist.gov/fpga.

Keywords: Photon statistics, Multichannel statistical processing, Quantum optics, Superconducting detectors.
PACS: 42.50.Ar; 85.25.Oj; 85.60.Bt.

INTRODUCTION

The growing interest in research and applications of photon-counting technology is clearly evident as more laboratories use single-photon technologies for applications, such as quantum communication, quantum computing, single-molecule monitoring, precision measurements, etc. To implement even a simple photon-counting test bench, one needs to invest heavily in not only single-photon detectors (SPDs), but also in expensive photon counting hardware and software. Even more problematic is the fact that most commercial solutions rely on proprietary (as opposed to open-source) software, making it difficult to adapt these solutions to custom needs, especially when real-time data processing is needed. A wide variety of instrumentation for pulse detection and characterization has been developed by nuclear physics community [1]. Although plenty could be learned from their experience, most of those detection devices are complex and/or not readily convertible to fit the needs of quantum optics. There have been other solutions featuring low cost and simplified design [2].

Recently, rapidly developing Field Programmable Gate Array (FPGA) technology has attracted a great deal of attention from scientists who need highly versatile and reliable measurement and data processing devices, including those that operate in extreme conditions (see Ref. [3] and references within). However, developing FPGA-based instruments in a physics laboratory can be a complex and time-consuming endeavor [4]. We present a simple data acquisition solution for both digital and analog single-photon detectors that is based on low cost, commercially available

Advances in Quantum Theory
AIP Conf. Proc. 1327, 505-519 (2011); doi: 10.1063/1.3567482

hardware and open-source software and firmware that can be compiled using free compilers. This paradigm allows the end user to easily tailor the proposed generic solution to specific applications and is particularly useful in devising one's own experiments with unique requirements.

We will first introduce a flexible platform used as a basis to implement a number of acquisition devices. We will then illustrate the versatility of this platform by describing the design of a digital and an analog pulse measuring board. We believe that this technology will find a broad range of applications in undergraduate and graduate physics laboratories and be useful tool in training the next generation of experimentalists.

OVERVIEW

In most photon-counting applications it is essential to convert signals generated by single-photon detectors into digital signals that are recorded by a computer in real time for detailed analysis. Many such applications use single-photon avalanche photodiodes (SPADs) that are not photon-number resolving, i.e. they generate a single electrical output pulse independent of whether one or more photons were incident. For these detectors, one can fully describe the output by recording the arrival time of the positive edge of each electrical pulse. To determine the arrival time of a pulse, one needs to establish a clock with a sufficiently high update frequency and record the most recent clock value upon arrival of the pulse. Ideally, the clock rate should be no slower than the inverse of the SPAD timing jitter (i.e., ≈1 GHz). But in many applications, especially involving pulsed single-photon sources, such high resolution is not required. In our implementation, we trade resolution for simplicity and aim at clock rates of >170 MHz (i.e. twice the repetition rate of typical ultrafast lasers). For multiple detectors, recording of the pulse edge arrival times should be done in parallel. In some cases, detections on multiple channels are correlated and only simple correlation measurements between channels are needed. For these cases the absolute arrival times of individual pulses do not need to be recorded. In other cases, it is important to know absolute arrival times. While collecting and storing absolute arrival times in a computer (as opposed to pre-processing data on an FPGA) is more difficult than storing correlations because of the real-time computer resources required, modern computers are often able to keep up with these needs for sufficiently low count rates. Also, if all the raw arrival time data are recorded, post-processing statistical analysis can be easily varied by simply editing computer software (i.e. without modifying an acquisition board and its firmware). Our solution allows for both types of instruments. As an example, we have chosen to present the second of these approaches where all timing data is sent to the computer to take advantage of the flexibility of having the data processed by computer software.

There are also detectors with analog outputs, some of which are photon-number resolving. There is no way to keep all the information encoded in an analog signal (unless of course a complete waveform is digitized and recorded, but that is often not practical). However, because we know the shape of the analog pulses expected from a single-photon detector with an analog output, we could characterize the pulse by recording just a few parameters: its height, duration and area.

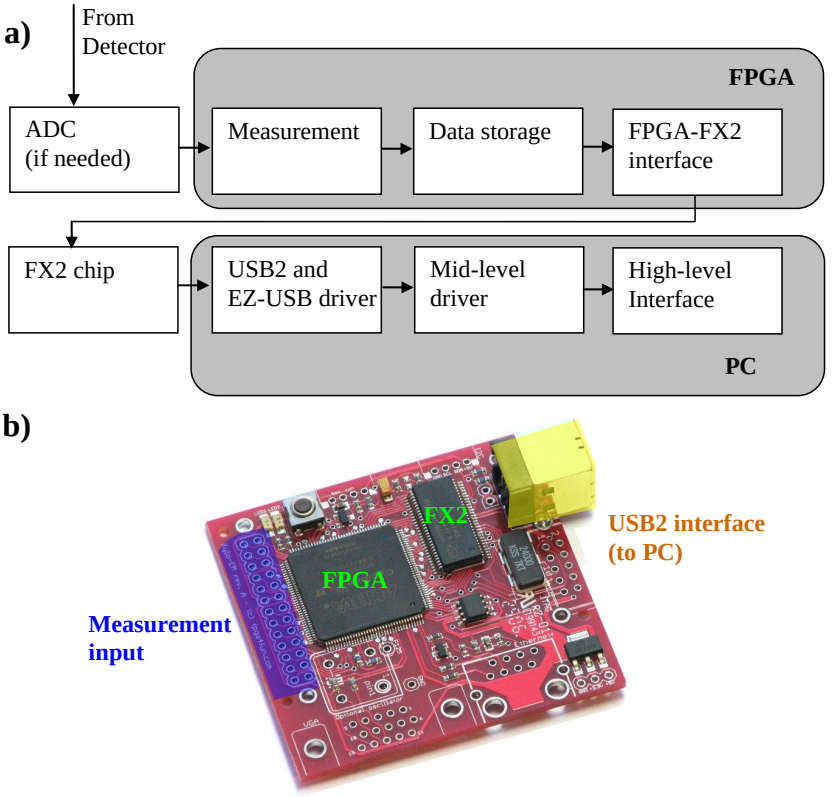

FIGURE 1. (a) Block diagram of the platform (data collecting, storage, FPGA-FX2 interface, FX2 multi-directional interface, EZ-USB driver, low-level PC software, high-level user (visualization) software. To record an analog signal, an Analog to Digital Converter (ADC) may be used. (b) Platform implementation using commercially available FPGA prototyping board "Xylo EM".

Therefore, an analog detection event is described by a finite series of discrete data that can be stored and transferred to a computer in a similar fashion as with digital detection. Because of the inherent flexibility of the modular structure of our platform, we only need to modify the analog monitoring/data generation part of the firmware, while leaving the storage and communication modules untouched.

PLATFORM

Two requirements of a data acquisition platform are a processing unit capable of making a measurement and a communication unit that sends acquired data to the computer. Because measurement blocks could significantly differ from one another, in our design (Fig. 1) we focused on developing a communication protocol that would

be suitable for a wide range of measurements on random electrical pulses. The universal serial bus USB-2 (Ref. [5]) interface is a popular and standard way of communicating with a computer, and generic USB-2 controller chips are readily available. Also, there exist open-source drivers for these chips that greatly simplify communication between a data-acquisition device and a user application. For implementation of a data-acquisition unit, one can use FPGAs because they are well suited to perform logical operations at high speeds and are also extremely versatile. So the same FPGA chip can be used for gathering data from both digital and analog detectors.

The design of a measurement block (Fig. 1(a)), as well as the format of data it generates, is determined by the specifics of the measurement, and will be discussed separately for each acquisition device. To reduce implementation costs, we use a commercial FPGA prototyping board, "Xylo EM," [6] as our data acquisition platform. It has both a USB-2 controller chip (Cypress CY7C68013A, also called FX2) and an FPGA chip (Altera Cyclone II) along with the FPGA program loading software [7] and more than 20 FPGA pins available to connect to external measurement electronics, Fig. 1b. The board is powered from a personal computer (PC) USB-2 port, so no external power source is required. As evident from Fig. 1, implementation of the platform requires only easy soldering. We provide tested FPGA firmware for the instruments discussed below. Because we distribute firmware with source codes, making custom instruments is greatly simplified. Once the board is programmed, a computer can communicate with the board via the EZ-USB [8,9] driver distributed with its source code. We developed and tested our system with a Windows XP computer, although the platform could be made to work under other operating systems as well, with a proper driver that supports an FX 2 chip. For instance, a libusb driver can be used with Linux and Mac OS. [10]

BOARD-TO-PC COMMUNICATION

The most essential component of this platform is the FPGA to PC interface that sends acquired data to the PC. Once established, this component can work with most instruments, regardless of the method of the measurements and data reduction required. More specifically, because the FPGA and FX2 chips are physically connected, and communication between the FX2 and PC is already established with a low level EZ-USB driver, we only need to program the FPGA to transfer data to the FX2 and write a simple mid-level driver for a PC that follows the data transfer protocol. Because the board is geared towards random event characterization, the transfer protocol should function regardless of how much data, if any at all, is available for each download cycle. The standard USB-2 protocol makes the PC a master and any external device a slave. Thus the board may only send data to the computer when the computer is ready to accept data. Also, the computer must know how much data it will download *before* initiating the download. We emphasize that it is impossible to predict how much data will be available for each download cycle, which often results in communication problems for the interface designer. On the other hand, using the prototyping software and firmware provided with our board deals with this issue, is very cost effective, and allows one to focus immediately on the

application at hand. To accomplish this while taking advantage of existing drivers and hardware, we use a two-step protocol. While understanding of this protocol is not required for implementing and operating the instruments or even to make minor modifications to those instruments, its description is important for the more advanced user who may want to build a more sophisticated tool and/or use other FPGA chips and evaluation boards.

A. FPGA Side

The FX2 stores data being transferred between the computer and the FPGA in First-In, First-Out queues (FIFOs) that can store multiple packets of data. Separate data streams use separate FIFOs. The FX2 can support up to four separate FIFOs. When a data transfer is initiated, using the EZ-USB [8] driver on the PC and the native prototyping board FX2 firmware, the FX2 accumulates a full packet of data in its FIFO before outputting it. Additional data is accumulated in the next packet of the FIFO, while it sends the previous packet. The FX2 on this board uses a default USB2 data packet size of 512 bytes of data. At low photon detection rates, when less than 512 bytes of data are ready when the PC requests data, the FPGA firmware must force the USB2 controller chip to send less than a full packet to the computer. The FPGA does this by asserting an end of packet flag on the FX2. Otherwise the USB2 packet will be delayed until all 512 bytes are collected, so that data updates will occur with somewhat random intervals. Applications requiring real-time updates of data with uniform intervals are often important (for example the alignment of optical detectors). If the length of the collected data is larger than 512 bytes, then one could send data in multiples of 512 byte packets and keep the excess for the next communication cycle. First, when the PC is ready to accept data it transmits a known size command packet to the FPGA, on FIFO_3, requesting the data size. Then the FPGA sends back a known size packet containing the data length to the PC on FIFO_2. Second, the PC sends another command packet to the FPGA requesting that amount of data. Then the FPGA sends that amount of data to the PC on FIFO_1. This two-step protocol is versatile and can be used for any task that generates random-length data. Other commands can be initiated by the PC, on FIFO_3, for example: starting and stopping acquisition, resetting the counter, setting thresholds, etc. The interface protocol part of the FPGA program is presented in Fig. 2.

B. PC Side (mid-level driver)

Custom PC software development for a data taking instrument is extremely simple, because it only has to deal with interpreting the information from the data acquisition board and leaves the low-level communication task to the EZ-USB driver. Because the low level software can initiate the communication protocol at any time, the information about the last event might get truncated. If this is the case, the remaining bytes of this event will be received during the next communication cycle. Hence, the mid-level driver must keep track of incomplete events and insert them back into the data "stream" as soon as they get completed. In a final step, the mid-level driver writes the raw data to a hard drive for post processing analysis and/or launches a real-

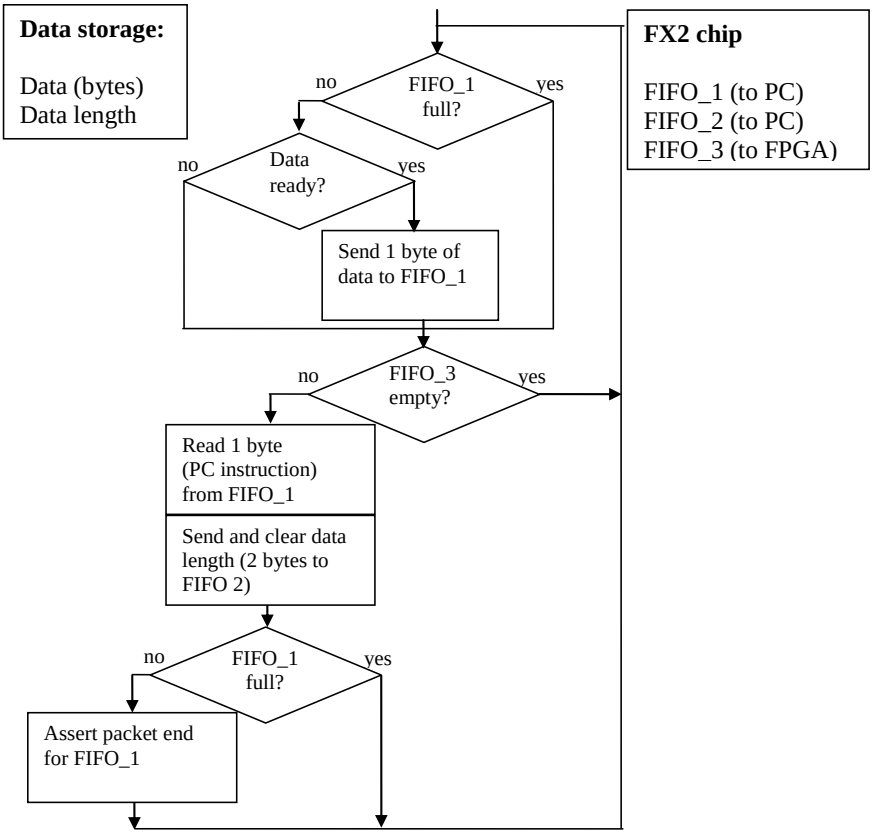

FIGURE 2. Communication protocol flowchart.

time statistical analysis function. A mid-level driver has three interface functions. FPGA_Open() opens a communication session and returns a handle to FPGA hardware. This handle is used for the duration of a communication session. FPGA_Close (HANDLE XyloDeviceHandle) closes the communication session. The main input/output function that communicates with an FPGA is: int FPGA_Interface (int runs, char fpga_command, int saveclicks, char * filename, HANDLE XyloDeviceHandle, int * length, unsigned char * data). On success, the function returns 1 and fills an array "data" with collected (and/or processed) data. If an error has occurred, the function returns a negative value or zero. The parameters listed in Table 1. If desired, this mid-level driver could be compiled as a dynamic link library (dll) and be accessed from high-level data acquisition and visualization software, such as LabView. For user convenience, example VIs that call the interface functions are provided for each instrument.

TABLE 1. Input/Output parameters for FPGA_Interface function.

Parameter	Input or Output	Description
runs	input	how many communication cycles does this function make before returning execution to the caller (LabView [11] by default).
fpga_command	input	one byte communication from PC to FPGA, used to start, stop acquisition, set thresholds, etc. Refer to a specific instrument manual for the list of acceptable commands.
saveclicks	input	if set to 1, saves the raw output from FPGA to a file otherwise it does not.
filename	input	name of the file to save raw output if Saveclicks=1.
XyloDeviceHandle	input	handle to an FPGA device
length	output	length of received data packet.
data	output	pointer to the data array.

Based on the method of data transfer described above, one can build various acquisition devices. In what follows, we describe how to build two devices that perform completely different measurements. Namely, we describe a four-channel coincidence detection instrument that uses digital TTL pulses and a one-channel analog-pulse characterization instrument. We invite the reader to build replicas of our acquisition boards to get started, and then make their own modifications of the measurement and/or data reduction blocks to meet the needs of their own experiments. Users can use Verilog [12] and/or a graphical language (i.e. a free edition of Quartus II in case of Altera FPGA chips) and upload their firmware to the FPGA. Xylo boards come with a Windows-based configurator that uploads firmware with a one button click. Note that Ref. 1 is a particularly good resource that describes various data acquisition protocols that could be implemented on our platform by only changing the code in a measurement block. We stress that creating and testing new instruments based on existing instruments built with this platform is easy enough that we envision a laboratory class where students design and apply a modification to optimize data acquisition based on the particular needs of the lab experiment at hand. For example, if the objective is to time-tag coincidences between channels A & B and A & C, because A acts as a trigger, a student could place the two "AND" gates in the program, compile and upload the modified code and observe the difference it makes relative to the unmodified software. Similarly, one could apply a numerical filter that selects pulses of a certain length before they get counted by the program. For this transformation, an input pulse becomes the input to a triggered summator whose output bit is asserted only if the sum gets larger than a certain value – another simple modification that can be done by a student during a laboratory session.
We stop here, leaving the joy of conceiving and implementing new instruments to the reader.

INSTRUMENT 1: MULTI-CHANNEL DIGITAL DATA AQUISITION

We now discuss how to configure our platform for a multi-channel digital application. The primary goal of this particular device is to collect timestamps of positive edges of

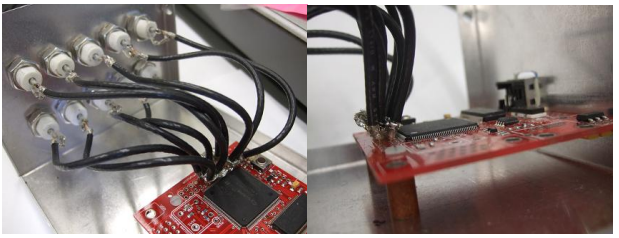

FIGURE 3. Mounting BNC connections and a board for Digital Multi-channel data acquisition.

pulses for either real-time or delayed statistical processing. Apart from the intended purpose, the device can measure velocity (i.e. time between the two events measured by sensors at a known distance), frequency, etc. We implemented the instrument with one start and four stop channels. An event on the start channel zeroes the clock and produces a record that is transmitted to the PC. A start channel event can be caused by a command from the pc, an external signal (e.g., the laser transmit clock) or counter register overflow. An event on any of the four stop channels, including simultaneous events (coincidences) produces a record with a timestamp (i.e. clock value) and the state of all 4 channels. The event is defined as the detection of a positive edge of a TTL pulse (at ≈1.4-1.5 V level). Once detected, the event is stored until the end of a nearest clock cycle of the instrument when the timestamp and the event detections of all 4 channels are recorded to a FIFO. The FPGA is then ready to detect another positive edge transition. This method guarantees that a maximum of one record is made per each clock cycle. Alternatively, one could timestamp events on each channel separately, thus keeping as many records as there were events per each clock cycle. The benefit of our design is threefold. First, we decrease the length of communication during coincidence bursts (at the expense of sending less information during low-count rate operation). Second, we reduce the time spent on analysis of coincidence events by the PC. Third, this design guarantees the absence of deadtime in the acquisition hardware, because the instrument is able to capture event information at each clock cycle. Hence, regardless of the history of preceding events, the registration logic is always ready for detection by the next clock cycle.

TABLE 2. Digital multichannel data acquisition instrument parts list.

Part	Quantity
Xylo-EM FPGA development board	1 ea.
Generic Electronics Box 3”x2”x6” (min)	1 ea.
USB-2 cable	1 ea.
BNC connectors	6 ea. (or more)
BNC cable	<1 m
Plastic mounting screws, bolts	As necessary

The parts necessary to build this instrument are listed in Table 2. It is evident (Fig. 3) that the total assembly required for this device is minimal. The physical parameters of the acquisition instrument for two-channel coincidence detection are listed in Table 3. The PC software can be written so that it groups time-bins together for the purposes of coincidence counting, hence the time-bin resolution will be the hardware

time-bins (given by timestamp increments). The FPGA based platform is fast enough to produce up to two time-bins per one typical mode-locked Ti:Sapphire laser pulse cycle that was used in our experiment and is typical of what is used to drive these single-photon systems. However, in some applications this timestamp increment can be a limiting factor, mainly because of the drift between unsynchronized clocks (i.e. board's internal and experiment's clocks) and detector jitter. To address this problem, we provide for an external signal from which an internal clock can be derived. This greatly reduces jitter by synchronizing all time increments to the main experiment.

TABLE 3. Digital multichannel data acquisition instrument Physical Characteristics.

Parameter	Value
Time counter reset (start) channel	1
Number of timestamping (stop) channels	4 (or more, after firmware modifications)
Timestamp increment (internal clock)	6.94 ns (144 MHz clock)
Minimal timestamp increment (external)	<6.5 ns (>160 MHz clock)
Board deadtime	None
Event threshold level (start, stop, clock)	TTL (<1.6 V, positive edge, not adjustable)
Highest USB data transfer rate	>2 10^6 pulses/sec (PC dependent)
Estimated cost	<$250
Estimated assembly and installation time	4 hours

TABLE 4. Data format.

Bit:	31	30	29	28	27	26-0
Meaning:	Counter cleared (by either "start" event, PC instruction or overfilling)	Event @ ch. 4	Event @ ch. 3	Event @ ch. 2	Event @ ch. 1	Timestamp

TABLE 5. List of BNC inputs.

Pin	Purpose
70	Detector (stop) 1
73	Detector (stop) 2
79	Detector (stop) 3
92	Detector (stop) 4
97	Clear the counter (start)
88	External clock

The full record (Table 4) of an event is 4-bytes (32 bits) long. The timestamp is stored in the lower 27 bits enabling continuous time-tagging for at least 0.8 s. The higher 5 bits represent events on each of the 4 data channels and the special, "start," channel that clears the counter.

To build the board, simply mount BNC inputs according to Table 5. We recommend wiring extra pins: 93, 100, 101 (and/or other) for future extensions and debugging. Then, using the configuration program that comes with the board, load appropriate firmware and start taking data with the VIs provided.

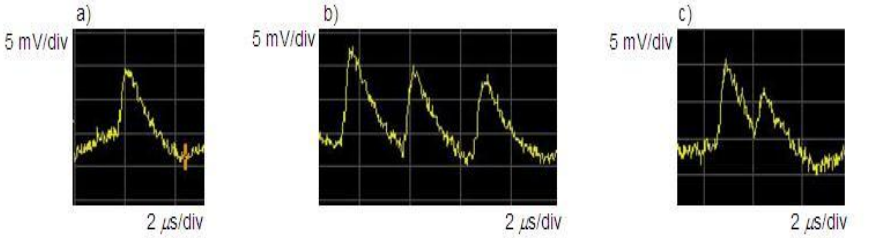

FIGURE 4. Typical analog output of a superconducting bolometer in response to a high level of continuous wave radiation. (a) single-photon detection trace, (b) multi-photon detection trace, and (c) multi-photon trace with two overlapped photon detection events.

INSTRUMENT 2: ANALOG DATA AQUISITION

A simple modification of the data collection module allows one to make an entirely different acquisition instrument based on the same acquisition platform. As an example here, we turn this platform into an analog data processing and logging device. Our design was inspired by the recent breakthrough in SPD technology based on superconducting bolometry [13,14], which has an advantage of nearly 100 % detection efficiency and photon-number-resolution, paired with the absence of afterpulsing and deadtime. A superconducting bolometer produces a continuous analog output that describes a physical condition of its optical element and needs constant monitoring. We designed the acquisition instrument to sample this detector at a ≈100 MHz rate, exceeding its typical output bandwidth. Then, our acquisition instrument applies data reduction algorithms only to characterize signal pulses and reject noise. To record the analog detector output, we use an external Analog to Digital Converter (ADC). If an 8-bit ADC is used, it would generate 100 MBytes/s of data that must be processed regardless of the single-photon detection rates.

Typically, the detection of a single photon results in a single pulse (Fig. 4(a)) whose dimensions (especially height) depend on the wavelength of a detected photon. This signal can be characterized by its height only. The train of pulses (Fig. 4(b)) also can be characterized the same way, if the monitoring circuit has no deadtime. Difficulties arise when counting multiple photons in a pulse such as the one shown in Fig. 4(c). In this trace, two photons are absorbed by the detector close in time to one another. Whatever voltage threshold is used, it will likely be set too low to register both pulses individually. In general, the output signal crossing the threshold does not necessarily mean that only one photon was detected, or that a photon detection took place at all. In addition, measuring the height of the pulse does not guarantee a certain number of photons were seen. Thus, additional parameters of the pulses must be recorded for accurate photon counting. This also becomes apparent when considering the interplay of signal counts with background noise, which is even more complex.

Although this algorithm was developed for superconducting bolometers, the problems in correctly interpreting the photoelectronic output is common to many number resolving single photon detectors. Particularly, low-cost multi-pixel photon

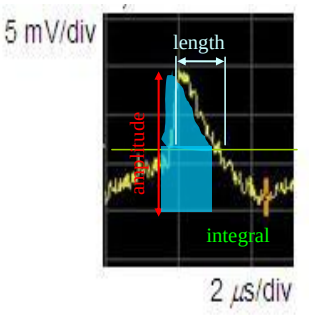

FIGURE 5. Principle of pulse characterization.

counters (MPPCs) that are commercially available (for example, Hamamatsu unit S10362-33-050c) can offer true photon number resolution, if their analog output is characterized in exactly the same way as it is outlined above.

Our algorithm is based on the known response of the detector to a single photon and the fact that the detector's response is (in a first approximation) the linear sum of all single-photon instrument functions. Based on this assumption, one can fully characterize the detector's output by recording height, length, and the integral under the pulse (defined with respect to some threshold), see Fig. 5. This way, every photon detection will be encoded in just a few (8 in our implementation) bytes. Our data format is shown in Table 6 and our algorithm is presented in Fig. 6. The advantages of our method are preservation of detector linearity, the ability to separate a detectors background noise from an optical signal, and the assurance of deadtime-free

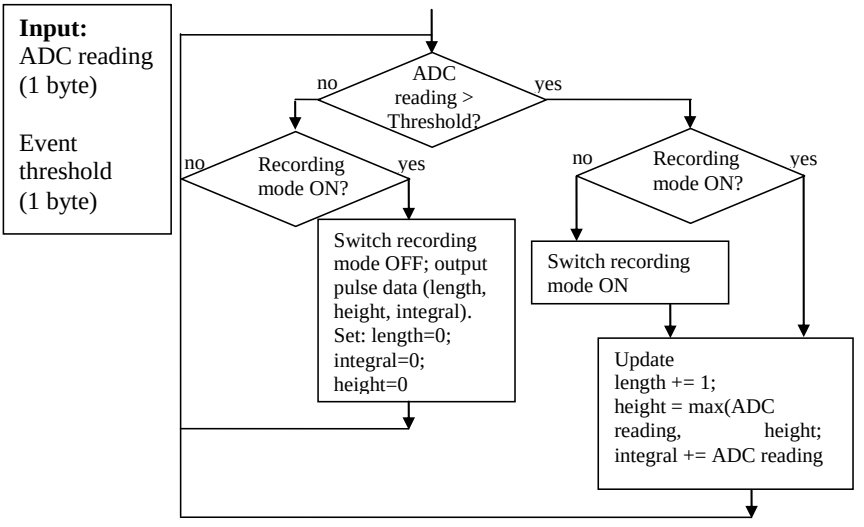

FIGURE 6. Analog pulse detection flowchart.

continuous monitoring of the bolometer's output. In particular, because of continuous monitoring of the input (no detection deadtime), the trace in Fig. 4(b) will be interpreted as three independent single-photon detections. The trace in Fig. 4(c) will produce an integral output that is consistent with detection of two photons; however its amplitude would be consistent with detection of a single photon. In this case, the independent measurement of a longer than usual pulse length, would indicate an overlap between two single-photon detections. The idea to characterize pulses by independently measuring several pulse characteristics is of course not new. Particularly, independent analysis of the amplitude of the process (to determine pulse presence by crossing a pre-determined threshold level) and the integrated signal (to discriminate between various pulse shapes) was implemented and used for nuclear physics experiments [15]. The difference here is that by taking advantage of a dedicated hardware module offered by our platform, we perform all data extraction and reduction digitally in a single chip, which simplifies the implementation and brings the cost down. Moreover, our platform allows for implementing other digital algorithms for signal extraction from an analog source. For a review of such techniques refer to Ref. [16].

Of course, our system can be used for any other analog input. Similar pulse characterization is required for many physics, electrical engineering and chemistry experiments. The board described here is well suited for situations where the shape of pulses is expected to be similar to one another while rate, peak amplitude, length, and integral of pulses generated by the underlying process are unknown a-priori. Possible applications include characterization of single dust particles or scattering centers in gases or liquids. It could also be used to monitor spikes on power or communication lines to detect anomalies and failures. This acquisition board can also be paired with high-energy particle detectors or it might be useful for characterizing periodic and quasi-periodic processes, such as monitoring a heartbeat.

TABLE 6. Data format.

Byte:	7	6-4	3-0
Meaning:	Pulse peak	Pulse duration (time bins)	Pulse Area

TABLE 7. Analog data acquisition instrument parts list.

Part	**Quantity**
Xylo-EM FPGA development board	1 ea.
Flashy rev. K development board	1 ea.
Generic Electronics Box 3"x2"x6" (min)	1 ea.
USB-2 cable	1 ea.
BNC connectors	1 ea.
Plastic mounting screws, bolts	As necessary

Possible modifications of this instrument include changing the sampling rate to match the intended analog input (the FPGA hardware is capable of sampling with any rate up to ≈100 MHz with the recommended ADC development board and up to ≈200 MHz if a different ADC is used with the specific FPGA at hand), adding a digital low-pass filter, or changing the number of ADC bits. Other real-time digital processing

features such as smoothing with running averages can be also added to the design, requiring in most cases, only minor changes of the firmware.

Our analog acquisition instrument is based on the same platform as our digital acquisition instrument, with the addition of a separate board with an 8-bit ADC chip and an integrated analog amplifier (Flashy rev. K development board, see Table 7). With the help of an integrated amplifier, the input signal can be aligned to fall into the interval between a few tens of mV to about one volt. This corresponds to a ≈5 mV digital resolution. The physical characteristics of the instrument are presented in Table 8 below.

TABLE 8. Analog data acquisition instrument physical characteristics.

Parameter	Value
Number of analog channels	1
Data sampling rate	100 MHz (example) <200MHz (custom designs)
Signal level	~10-1000 mV
Typical detectable voltage difference	~5 mV
ADC width	8 bit
Board deadtime	None
Event threshold level	adjustable (08h - F8h with 08h increments) voltage is implementation dependent
Highest USB data transfer rate	$> 10^6$ pulses/sec (PC dependent)
Information acquired	Pulse peak, duration and integral
Estimated cost	< $350
Estimated assembly and installation time	6 hours

TABLE 9. List of ADC-FPGA connections.

FPGA Pin	Flashy (ADC) Pin	Purpose
132	13	Data 0 (ADC->FPGA)
133	14	Data 1 (ADC->FPGA)
114	8	Data 2 (ADC->FPGA)
113	6	Data 3 (ADC->FPGA)
112	5	Data 4 (ADC->FPGA)
97	3	Data 5 (ADC->FPGA)
104	2	Data 6 (ADC->FPGA)
100	1	Data 7 (ADC->FPGA)
88	7	ADC clock (FPGA->ADC)
Any pin marked ground	4, 10 or 16	Ground (FPGA->ADC)
Any pin marked +3.3V	11, 12	+3.3 V Power (FPGA->ADC)

Our preliminary testing shows that the analog acquisition board and the PC is capable of real-time statistical analysis of up to 1 million detection events per second, matching the maximum counting rate of current, state-of-the art photon counting bolometers. The performance of this algorithm on a bolometer output has been recently reported [17].

To build the instrument, connect the two boards following Table 9, attach an analog input BNC connector to the Flashy board and load the pre-compiled firmware. Adjust potentiometers of the Flashy so that the range of the ADC matches the signal source.

The Flashy board generates its own clock at 100 MHz that is used for sampling and synchronization with FPGA. To set a different data sampling rate on a Flashy rev. K development board, follow the instructions supplied with that board. To use a different ADC, use Table 9 as a guide on how to connect it to the FPGA board. The pre-compiled example needs an external clock synchronized with an ADC. Deriving a clock signal from an FPGA chip and feeding it to an ADC requires small changes in the FPGA firmware to produce the clock and synchronize the data-collecting part with it instead of using an external source, but requires no changes to drivers or the USB2 protocol. The sample Labview code collects information about events and builds three histograms for each parameter collected.

OTHER ACQUISITION INSTRUMENTS

One can use this same platform (with appropriate modifications) to develop other data acquisition instruments. For example, if timestamps are not critical for the application, statistical processing could be done on the FPGA chip. This significantly reduces data transfer to a PC. Such an instrument was also implemented and used for higher-order coincidence measurements by the authors [18], but that instrument's description is beyond the scope of this manuscript.

CONCLUSION

We have presented an inexpensive fast-pulse characterization platform capable of real-time operation, suitable for acquisition of single-photon data. Based on this platform, we developed both a digital multi-channel data acquisition instrument and an analog pulse acquisition instrument that should find wide application in undergraduate physics laboratories. The digital multi-channel acquisition instrument provides performance comparable to more elaborate commercial instruments (although the timestamp increment of commercial boards may be an order of magnitude shorter) for a fraction of the cost. It is simple to assemble and install and has open-source code allowing easy customization. The analog data acquisition instrument is capable of continuously monitoring SPDs with an analog output. By design the analog data-acquisition instrument does not introduce any deadtime of its own and can be used in conjunction with a detector that has no deadtime for deadtime-free single-photon detection. A complete description of the platform, including source codes and most current updates are available for download from a project resource site http://physics.nist.gov/fpga.

ACKNOWLEDGMENTS

Authors thank Alan Mink for fruitful discussions and introduction into FX2 chip programming.

REFERENCES

1. G. F. Knoll, *Radiation, Detection and Measurement*, New York: John Wiley & Sons, 1979, 2000.
2. D. Branning, S. Bhandari and M. Beck, *Am. J. Phys.* **77**, 667-670 (2009).
3. B. Aparicio del Moral, J. Rodríguez-Gómez and A. C. López Jiménez, *Scalable Computing: Practice and Experience* **8,** No. 4, i-iv (2007).
4. H. Xu, L. Ma, A. Mink, B. Hershman, and X. Tang, *Optics Express*, **15**, 7247-7260 (2007).
5. http://www.usb.org/
6. Certain commercial equipment, instruments or materials are identified in this paper to foster understanding. Such identification does not imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the materials or equipment are necessarily the best available for the purpose.
7. http://www.knjn.com/docs/KNJN%20FX2%20FPGA%20boards.pdf
8. http://www.cypress.com/design/DK10020
9. FPGA development board, and our firmware are compatible with CyUSB, a next generation driver available from Cypress: http://www.cypress.com/?rID=34870. To use CyUSB, PC software needs to be modified. Note that the source code for cyusb.sys is not available.
10. www.libusb.org Note that to use this driver, PC software needs to be modified.
11. http://www.ni.com/labview/
12. IEEE Standard for Verilog® Hardware Description Language IEEE Computer Society IEEE Std 1364™-2005 New York
13. A. J. Miller, S. W. Nam, J. M. Martinis, *Appl. Phys. Lett.* **83,** 791-793 (2003).
14. S. Nam, J. Beyer, G. Hilton, K. Irwin, C. Reinstema, and J. M. Martinis, *IEEE Transactions on Applied Superconductivity* **13,** 618-621 (2003).
15. V. T. Jordanov, G. F. Knoll, *IEEE Transactions on Nuclear Science* **42,** 683-687 (1995).
16. T. H. Wilmshurst, *Signal Recovery from Noise in Electronic Instrumentation*, Second Edition, London: Taylor & Francis Ltd, 1990.
17. A. J. Pearlman, S. V. Polyakov, A. Migdall, Sae Woo Nam, *CLEO/QELS*, JThS56 Baltimore, MD, 2008.
18. E. A. Goldschmidt, M. D. Eisaman, J. Fan, S. V. Polyakov and A. Migdall, *Phys. Rev. A* **78**, 013844 (2008).