Nuclear Instruments and Methods in Physics Research A 701 (2013) 7–16

Contents lists available at SciVerse ScienceDirect

Nuclear Instruments and Methods in Physics Research A

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

The FPGA Pixel Array Detector

Marianne S. Hromalik [a,b,*], Katherine S. Green [b], Hugh T. Philipp [b], Mark W. Tate [b], Sol M. Gruner [b,c]

[a] Computer Science Department, The State University of New York at Oswego, Oswego, NY 13126, USA
[b] Laboratory of Atomic and Solid State Physics, Cornell University, Ithaca, NY 14853, USA
[c] Cornell High Energy Synchrotron Source, Cornell University, Ithaca, NY 14853, USA

ARTICLE INFO

Article history:
Received 5 June 2012
Received in revised form
28 August 2012
Accepted 5 October 2012
Available online 12 October 2012

Keywords:
X-ray Pixel Array Detectors
Field Programmable Gate Array
Real-time data processing
Autocorrelation function

ABSTRACT

A proposed design for a reconfigurable x-ray Pixel Array Detector (PAD) is described. It operates by integrating a high-end commercial field programmable gate array (FPGA) into a 3-layer device along with a high-resistivity diode detection layer and a custom, application-specific integrated circuit (ASIC) layer. The ASIC layer contains an energy-discriminating photon-counting front end with photon hits streamed directly to the FPGA via a massively parallel, high-speed data connection. FPGA resources can be allocated to perform user defined tasks on the pixel data streams, including the implementation of a direct time autocorrelation function (ACF) with time resolution down to 100 ns. Using the FPGA at the front end to calculate the ACF reduces the required data transfer rate by several orders of magnitude when compared to a fast framing detector. The FPGA-ASIC high-speed interface, as well as the in-FPGA implementation of a real-time ACF for x-ray photon correlation spectroscopy experiments has been designed and simulated. A 16×16 pixel prototype of the ASIC has been fabricated and is being tested.

1. Introduction

The development of new ultra-fast, ultra-bright X-ray sources is creating many new experimental opportunities. Fully exploiting these opportunities requires detectors that are designed to complement the capabilities of these sources [1]. This drives the development of pixel array detectors (PADs) (Fig. 1) that take advantage of direct conversion of x-rays in semiconductors, which provides high spatial and temporal resolution with good signal-to-noise ratio [2–5]. Additionally, the ASIC layer offers a flexible platform for development of in-pixel signal processing.

Offline computer-based post processing of the acquired data requires that the massively parallel PAD output be serialized and formatted for readout before new data can be acquired. This results in a bottleneck at readout, significantly reducing the speed of imaging while simultaneously producing a potentially voluminous amount of stored data, especially for time resolved experiments [6]. On-chip data storage has successfully been used for very high speed applications [3,7] but in-pixel memory is limited. Increasing the complexity of the ASIC pixels to allow for real-time processing would result in complicated and potentially larger pixels. Most importantly, this would lead to PADs specifically tailored to particular experiments. While this has merits in some cases, the developmental lead time for PADs is on the order of 5 years, making it impractical and expensive for many applications.

The FPGA PAD seeks to solve this problem by adding a re-programmable layer to the existing PAD structure (Fig. 1). This fundamentally changes the data collection paradigm by providing high-bandwidth, low-level data processing that occurs within the detector prior to readout. The FPGA PAD thus consists of a traditional two-layered PAD with fast, simple photon-counting pixels outputting single bits at high speeds through a massively parallel interface to an FPGA. This makes the FPGA an integral resident part of the detector, rather than just a part of the control hardware.

By processing data in real time, the rate at which data is transferred to storage is reduced dramatically while the desired information content is efficiently extracted from the measurement. The FPGA can be reprogrammed to meet the needs of various applications, effectively creating a functionally new detector without the need to redesign the ASIC. This reduces the cost and lead-time required to adapt the detector for use in other experiments. A preliminary design of the FPGA PAD is presented with an example implementation of a real-time autocorrelation calculation.

2. System description

The FPGA layer consists of a high-end Xilinx Virtex6 or Virtex7 FPGA with a large I/O pin count that mates to the ASIC layer. The FPGA contains a sequential logic control unit (state machine) and

* Corresponding author at: Computer Science Department, The State University of New York at Oswego, Oswego, NY 13126, USA.
E-mail addresses: marianne.hromalik@oswego.edu (M.S. Hromalik), ksg52@cornell.edu (K.S. Green), htp2@cornell.edu (H.T. Philipp), mwt5@cornell.edu (M.W. Tate), smg26@cornell.edu (S.M. Gruner).

0168-9002/$ - see front matter © 2012 Published by Elsevier B.V.
http://dx.doi.org/10.1016/j.nima.2012.10.024

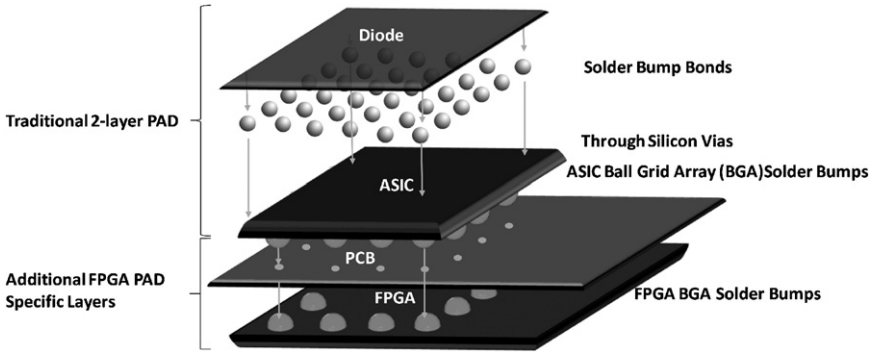

Fig. 1. Conceptual design of the FPGA PAD. A traditional PAD consists of a diode layer bump-bonded to an ASIC processing layer. The pixelated diode layer converts x-rays into charge. In the case of the FPGA PAD, photon hits are digitized into single bits in the ASIC and streamed through a massively parallel interface directly into the FPGA pins for processing. Through-silicon vias would enable high pin count connections between a large ASIC and the FPGA. The initial prototype detector does not require this feature.

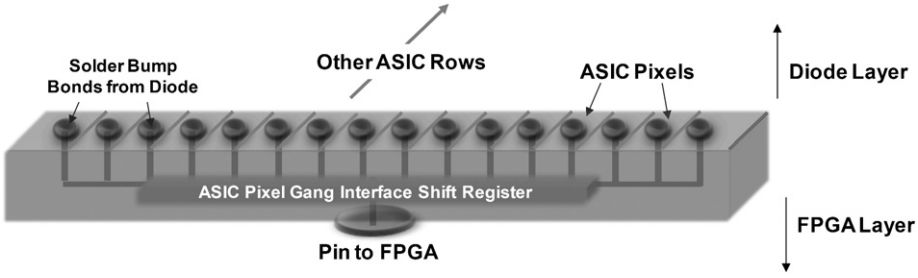

Fig. 2. Pictorial representation of a single pixel gang within the ASIC layer of the 16×16 FPGA PAD. The outputs of all of the 16 pixels in a row are fed into the pixel gang interface and streamed out serially to the FPGA.

interface design to handle the read-out of data from the ASIC. The remaining FPGA resources can be used to configure application-specific computation. A commercial FPGA was chosen as the back layer to enable development of application specific configuration modes by the end-user.

The interface between the ASIC and the FPGA layer determines both the speed and the degree of flexibility of the detector. Because the number of pins available for I/O on a commercial FPGA is limited, there cannot be a one-to-one mapping of ASIC pixels to FPGA pins. Instead, one FPGA pin is the I/O point for a group of pixels, hereby referred to as a pixel "gang" (Fig. 2). The readout of each pixel gang is event-driven, meaning that a gang of pixels is only read if one of the pixels in the gang detects an x-ray photon. The pixel gangs operate asynchronously and independently. The close physical proximity of the ASIC and FPGA layers reduces signal degradation, cross-talk, and transmission line effects associated with longer traces. This results in a highly parallel, asynchronous interface running at very high speeds.

Specifications for the FPGA PAD are shown in Table 1. The PAD is characterized by the fast x-ray count rate, the broad energy range of operation and the ability of the FPGA PAD to function as a high-speed imager. A full detector is envisioned to be constructed from 128×128 pixel tiles, each of which will be mated to a single Virtex7 FPGA. Each pixel gang consists of 32 pixels connected to one of the 512 FPGA I/O pins. The high pin count in this configuration requires the usage of through-silicon vias in the ASIC layer.

3. 16×16 Prototype

A 16×16 pixel prototype FPGA PAD ASIC has been fabricated to test the performance of the pixel front-end, several different configurations of the interface between the ASIC and the FPGA layer, and the functionality of the first example application: an in-detector autocorrelation function for XPCS experiments [8]. The only layer that is application specific in this description is the FPGA layer which is shown configured to calculate an autocorrelation function in real time.

Table 1
Design goals.

X-ray count-rate	10 MHz/pix
Detector layer	300–500 μm thick silicon
Quantum efficiency	> 98% (at 8 keV)
Energy range	2–20 keV
Tile array format	128×128 pixels (using through-silicon vias)
Pixel size	$150\ \mu m \times 150\ \mu m$
Pixels per ganged output	32 (16 for prototype)
Imaging modes	4 bits/pixel @ 625 kHz frame rate 10 bits/pixel @ 10 kHz frame rate 16 bits/pixel @ 150 Hz frame rate
Time autocorrelation function range	40 point log scale 100 ns–1 ms in FPGA. > 1 ms to use concurrent imaging mode for off-chip ACF.

3.1. Diode layer and XPCS

The $150\ \mu m \times 150\ \mu m$ pixel size in the prototype was chosen to utilize existing diode parts during initial prototyping. This size also allows ample space for pixel electronics, and matches the pixel gang area to the FPGA pin density.

In the longer run, smaller pixel sizes are often desirable to better match the speckle size with the pixel and achieve the maximum contrast in the speckle signal. Detailed analysis of contrast and SNR is needed to determine the best setup for a particular experiment [10]. Though the solid angle covered by a pixel can be decreased by increasing the distance between sample

and detector, existing beamlines are often constrained in this regard [9].

To reduce the effective size of a large PAD pixel, pixelated x-ray opaque masks have been proposed with an aligned hole over each pixel. Alternatively, a "double-bonded" diode pixel (Fig. 3) could be used where the diode pixel has two bump-bonds connecting it to its corresponding pixel in the ASIC layer with appropriate switching circuitry. Charge could be collected from either the smaller central region or the full area. There is a ca. 20 μm wide charge sharing region at the boundary of the sub-pixel. Charge sharing is generally not desirable if one seeks to measure the x-ray spectra, but for monochromatic radiation as would typically be used in photon correlation spectroscopy, setting a threshold at 50% of the incident x-ray energy will record all x-rays falling within the smaller sub-pixel. Further, one could adjust the effective size of the sub-pixel by increasing or decreasing the threshold from the nominal 50% value.

3.2. ASIC layer

A 16×16 pixel prototype of the ASIC layer was fabricated through the MOSIS service using the 0.25 μm TSMC 5-metal mixed-mode process using thick-oxide 3.3 V transistors. The majority of NMOS transistors use radiation-hardened linear techniques [11]. It includes a 16×16 pixel array, with 16 gangs for testing in conjunction with the data analysis FPGA, and a 17th row with physical test points incorporated within the pixels.

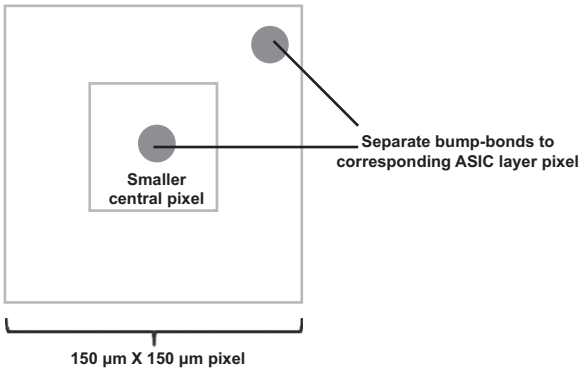

Fig. 3. Double bonded diode pixel. Switches within the ASIC layer pixel connect as input either the central smaller pixel alone or in combination with the larger surrounding diode area. The larger surrounding area would be shunted to a reference voltage if not connected as input.

3.2.1. The pixel

Fig. 4 shows the single-pixel schematic, comprised of the integrator, DC feedback circuit, low and high energy comparators and in-pixel latches. The DC feedback circuit compensates for dark current in the sensor [12]. A small bleed current is mirrored into each pixel by M1 and M2 to compensate for upto 1 pA leakage current in the sensor. The integrator is a differential-input folded cascode. In simulation, the sensitivity is found to be 26 μV/e− with a power draw of 97 μW per pixel. A 3-bit trim circuit adjusts the threshold of the comparator in each pixel. Increasing the number of trim bits will be addressed in future submissions. Fig. 5

C_F is chosen to be 5 fF to achieve good single-photon gain and is implemented as a bulk-connected PMOS device. The reset (RST) signal is individually generated for each pixel after gang readout is initiated, such that a pixel is only reset if it has registered a photon. Since resetting the integrator involves a small amount of charge injection, this selective reset avoids perturbing the input of pixels that have not registered any hits.

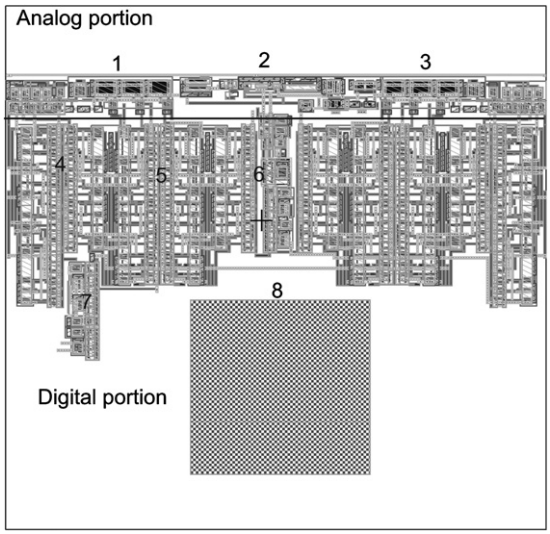

Fig. 5. Single pixel layout of prototype ASIC. Analog components, at top, include the integrator (2) flanked by high-energy (1) and low-energy (3) comparators. The remaining area is occupied by the bond pad (8) and digital components: in-pixel latches (4), 8 bits of in-pixel memory for trim and test bits (5), pixel reset logic (6), and pixel register logic (7).

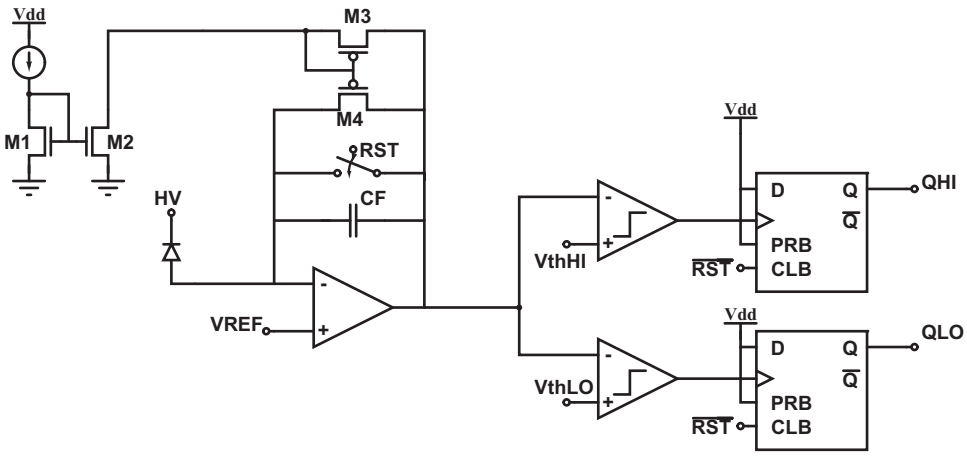

Fig. 4. High level schematic of a single pixel in the 16×16 FPGA PAD prototype. In single-bit mode, it produces a one-bit output indicating the presence or absence of a photon at the front end which exceeds a settable threshold. In two-bit mode, one of two bits is set if the photon energy exceeds a lower threshold. The second bit is set if the high threshold is exceeded.

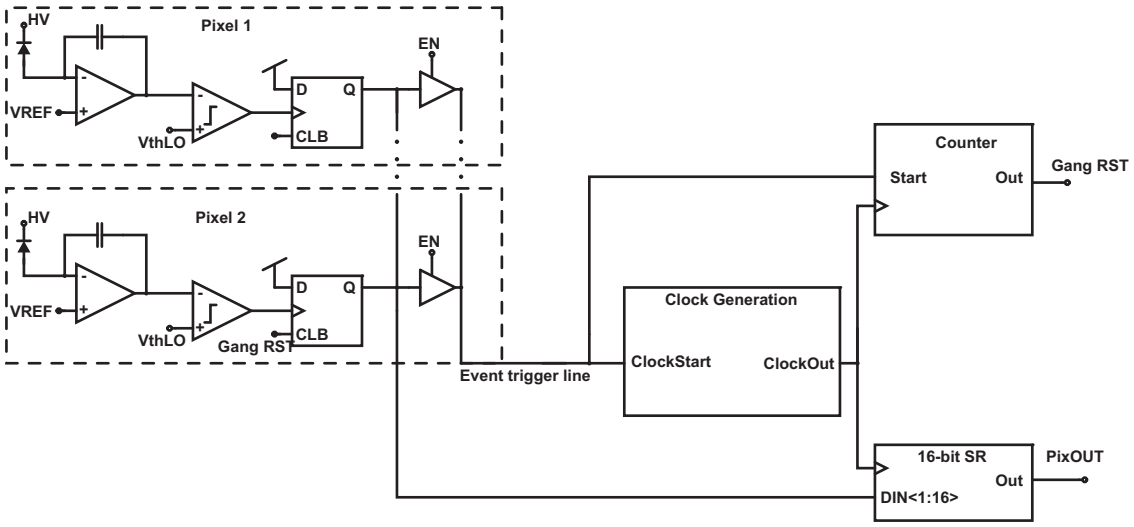

Fig. 6. The ASIC side of the ASIC layer/ FPGA layer pixel-gang interface. A photon hit at any pixel triggers a readout of the gang. Any photon hits which occur within the readout period are buffered in the in-pixel latch. This design allows data to be transferred to the FPGA with no dead time at a time resolution of 100 ns.

3.2.2. Data handling

Pixels are ganged together by row for communication with the FPGA. In the prototype ASIC, a gang consists of 16 pixels and a data handling block. A simplified block diagram is shown in Fig. 6. The high-energy comparator and latch are omitted. The output of the low-energy in-pixel latch for all pixels in the gang is fed via tri-state logic onto an event trigger line. When a photon event is registered by any pixel, the event trigger line goes high and initializes a readout sequence. The clock generation block contains a voltage controlled oscillator (VCO) which is disabled until gang readout is initialized. The user can also choose to route an external clock signal to all gangs. On readout initialization, a counter is triggered. After one counter cycle the gang reset signal is generated. This resets all integrators, in-pixel latches and data handling elements to prepare for the next incoming photon. A single read cycle will take $\sim$100 ns. As long as the incoming photon rate is not greater than 1 photon/100 ns/pixel, every photon incident on the diode and counted in the ASIC layer will be transferred to the FPGA for processing. The detector has no dead-time for readout and can discriminate photons arriving at different times to a minimum resolution of 100 ns.

There is also an option, not shown in the simplified diagram, to read out the high-energy comparator data in addition to the low-energy data, at the cost of doubling the readout time.

3.2.3. Testing ASIC electronics

Two methods of testing the electronics of the 16×16 array are built into the ASIC. To test the basic functioning of the data handling and readout, a digital test pattern is loaded into the pixel latches and a readout cycle is triggered by an external signal. To test the complete system including the pixel front-end, an analog test pattern is generated by use of an in-pixel charge injection circuit. The amount of charge injected is controlled by a global analog voltage and the duration of a digital charge inject pulse. Charge inject pulses are applied along pixel columns, perpendicular to the gang direction, so that only 16 bond pads are used but each pixel in a gang can have a unique test signal applied. Additionally, the prototype chip can be bump-bonded to a diode array so the bonded unit can be tested with x-rays.

3.3. The FPGA layer

The FPGA layer in the prototype consists of a Xilinx Virtex6 550T FPGA with 16 input pins closely connected to the ASIC layer output pins. Each of these pins is an interface to one of the 16-pixel gangs between the ASIC and the FPGA. Since all the pixel gangs are independent, there is a controlling state machine for each input data pin on the FPGA. This interface is application independent and will remain the same for all processing configurations of the FPGA layer.

The communication between a pixel gang and the FPGA is diagramed in Fig. 7. Detection of a photon event anywhere within the gang triggers an event-driven readout cycle. The interface pin of the gang sends a rising edge trigger followed by a bitstream with ones in the positions of pixels receiving photons. The FPGA input state machine deserializes the input pattern to determine which pixels have been hit from the position of the ones in the bitstream.

Two clocking methods are used for transmitting data from the ASIC to the FPGA. The first uses a tunable oscillator built into the data handling portion of each pixel gang. These oscillators are set to the same frequency as internal oscillators built within the FPGA fabric. This is an elegant, low-power, and potentially high-speed (upto 500 MHz) solution which requires no master clock. The oscillators must be tuned to each other—a task that may become tedious for large numbers of pixel gangs. Also matched oscillators on both the FPGA and the ASIC may drift in frequency with temperature. The second method uses an external clock to buffer and read out the data. This method, though simpler, is slower than the fastest speed possible with the matched oscillators and is susceptible to clock skew.

3.4. Example application—Autocorrelation for XPCS

One experimental application which illustrates the advantages of the FPGA PAD is x-ray photon correlation spectroscopy (XPCS). This technique exploits the interference of a coherent beam to study the dynamics of a sample. Correlations within the sample produce a unique pattern of speckles. Observing how these speckles change yields information about particle dynamics. In the appropriate geometry, a 2-dimensional detector array can be used to observe the time variation of speckles by identifying appropriate regions of interest in the scattering and tracking pixel outputs. The pixel output yields a measurement of incident x-ray intensity varying as a function of time. A description of this time-varying pixel output is given by second order coherence g^2, also

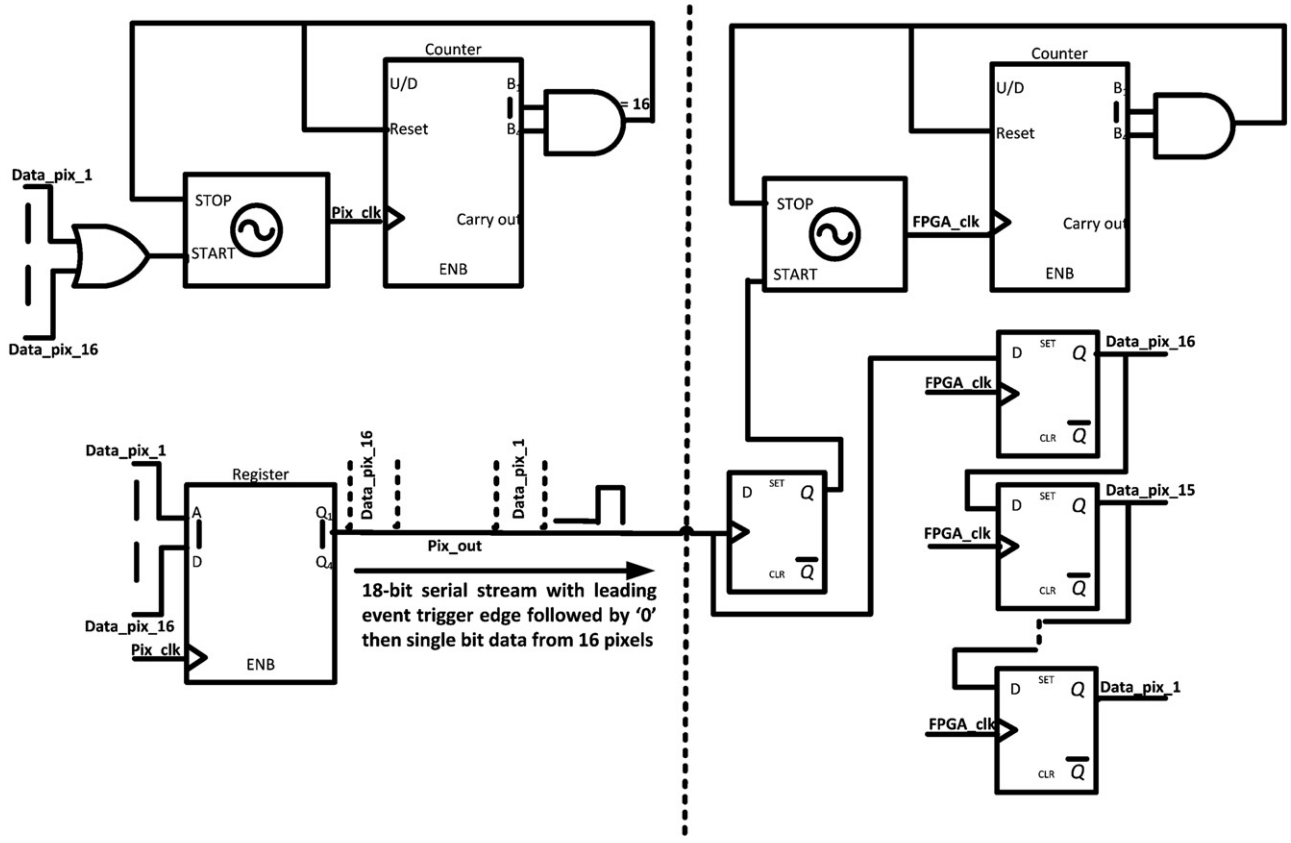

Fig. 7. Interface between a 16-pixel gang on the ASIC layer and the FPGA layer input pin. The one-bit pixel readings are serialized for transmission to the FPGA and deserialized within the FPGA interface. Each pixel gang interface is asynchronous and independent of the others.

referred to as the time autocorrelation function (ACF) [8]

$$g^2(q;\tau) = \int_{t=0}^{\infty} I(t)I(t+\tau)dt \quad (1)$$

A detector for XPCS should have adequate spatial resolution, the ability to distinguish single photons, and sufficient time resolution to study dynamics of interest [14]. It is also desirable to have a detector that covers a large solid angle [13]. The combination of these requirements (effectively requiring large and high-resolution coverage of signal, time and space) implies an information rate which is demanding for a typical framing detector (defined as a detector that produces images by digitizing all pixels, and conveys this information to be stored on some medium). The time scales of dynamics being studied by a framing detector are limited by the continuous frame rate.

The FPGA PAD addresses this problem by real-time processing of incoming pixel data. A discrete form of the ACF is calculated as the pixels respond to single incoming photons

$$g^2(q;j\Delta t) = \sum_{i=1}^{N} X_i(t)X_i(t+j\Delta t) \quad (2)$$

where $X_i(t)$ and $X_i(t+j\Delta t)$ are both either 0 or 1. This is calculated within the FPGA backend and the processed ACF data is then transferred to the controlling computer. The time resolution is not limited by the frame rate, but by the gang response time.

3.5. FPGA layer implementation of the ACF

A conceptual implementation of a four-point delay line ACF calculation is shown in Fig. 8 for a single pixel output [15]. The pixel output is clocked in by the master clock and compared to its previous values over 4 different delays. When there is coincidence (correlation), the relevant counter is clocked. The outputs of these counters are the four ACF values.

Using this scheme for, say, 10,000 time points from 100 ns to 1 ms is not achievable for a 128×128 pixel array due to limited resources within the FPGA. A system of levels is used instead to calculate the ACF within the FPGA.

3.5.1. Level 1 ACF calculation

The first level calculates the ACF using the streaming bitwise sum of products method from Eq. (2). Since the interface with the ASIC layer combines 16 pixels into 1 FPGA input pin, each 16-pixel gang is calculated using the same chain of latches. The input bit of each of the 16 pixels is therefore processed sequentially. The shift registers built in to the slices of the Virtex6 FPGA (SRL32CE) are used to minimize area usage and the ACF for each pixel for each time delay ($j\Delta t$) is accumulated and stored as a running 32-bit sum in on-chip block RAM (Fig. 9). This is implemented for each time delay from 100 ns to 1 μs for all the pixels on the 16×16 prototype. This uses less than 2% of the Virtex-6 FPGA resources.

3.5.2. Level 2 ACF calculation

While time delays ranging from 100 ns to 1 s are desirable in time-resolved experiments because they measure dynamics over a long range of time, resolution of 100 ns is not required for the entire time-scale. Sampling on a log scale with 10 time delays per order of magnitude may be an appropriate distribution of resources. To achieve this, the signal would ideally be sent through a delay line, with its fundamental delay element

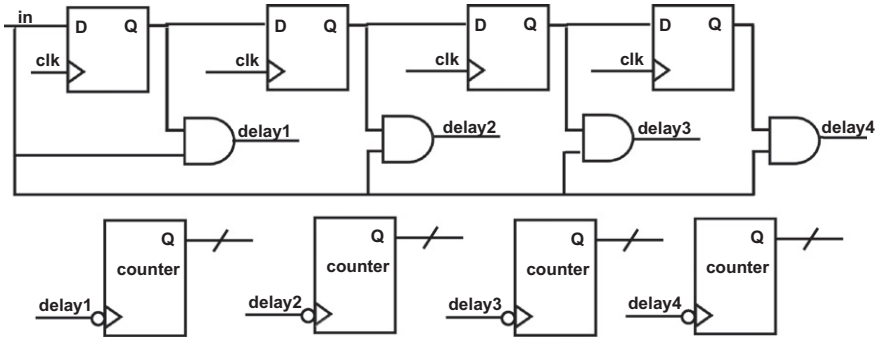

Fig. 8. Conceptual implementation of delay line based single bit autocorrelation function within an FPGA.

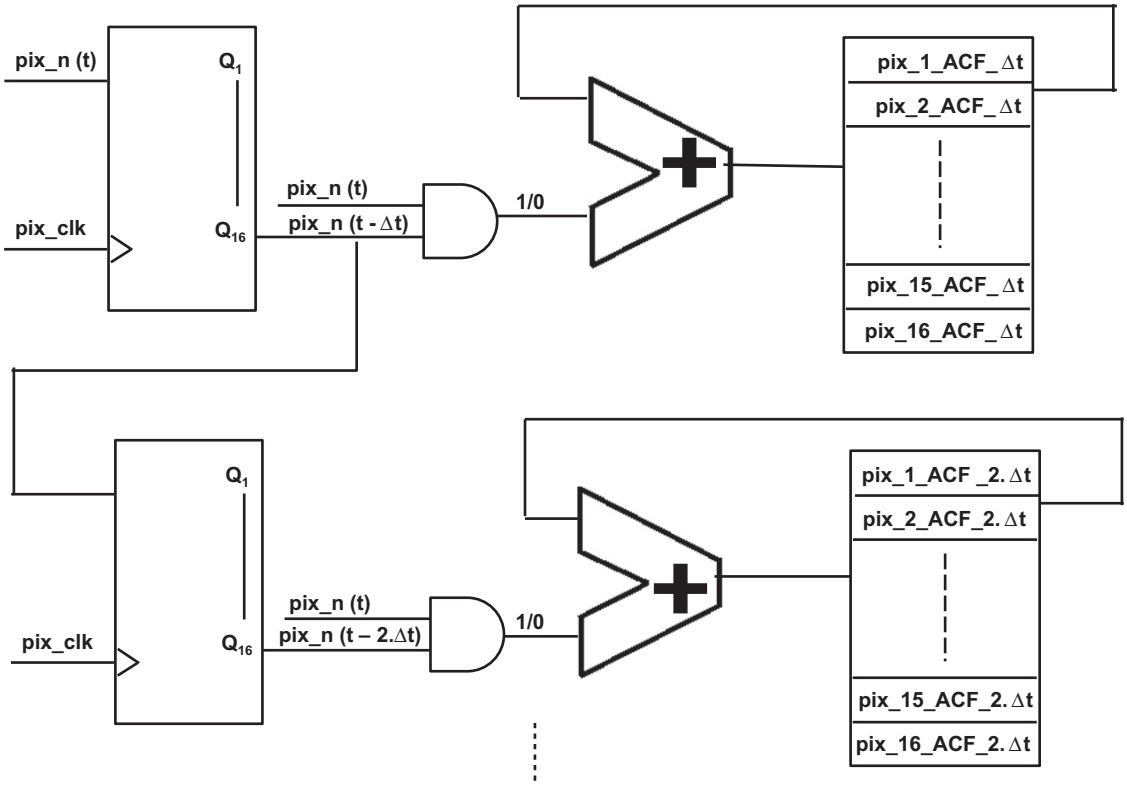

Fig. 9. Level 1 ACF calculation—real implementation of the delay-line single-bit ACF within the FPGA. The slice registers were used as the delay line to minimize area usage and a combination of an accumulator and on-chip Block RAM replaced the counters.

increasing in size logarithmically and the coincidences captured at appropriate intervals. To mimic this, a system of FIFOs was used to buffer the incoming 100 ns pixel bitstreams and delay them (Fig. 10). FIFOs of 10 positions long were used as buffers to realize a log scale ACF from 1 μs to 10 μs. 100-word long FIFOs were used as buffers for the 10 μs to 100 μs intervals and 1000 word FIFOs were used for 100 μs to 1 ms intervals. This resulted in a 30-point ACF over the range of 1 μs to 1 ms and used $<20\%$ of the XC6VLX550T Block RAM resources.

3.5.3. Level 3 ACF calculation

In the 16×16 pixel prototype, an accumulator sums the number of photons per pixel every 1 ms. This data is stored in on-chip RAM and can be transferred to the controlling computer for offline computation of the ACF for times greater than 1 ms. At this level, the FPGA PAD acts as a 13-bit photon-counting imager framing at 1 KHz.

While this functionality is useful in different applications and does not produce voluminous data for a 16×16 pixel array, performing the ACF calculation on-chip will prove very useful to reduce the amount of data transferred for a 128×128 pixel array. Full microprocessors can be implemented directly into the fabric of the FPGA logic (e.g., Xilinx core MicroBlaze). Using FPGA-based microprocessors to calculate the ACF for slower time-lags will be investigated.

4. Testing of ACF application on FPGA backend of 16×16 prototype

The functionality of the interface and the FPGA processing layer was tested using the Xilinx Simulator using both functional and timing simulations.

16 parallel input bitstreams were fed into the testbench interface from data files generated in MATLAB representing test

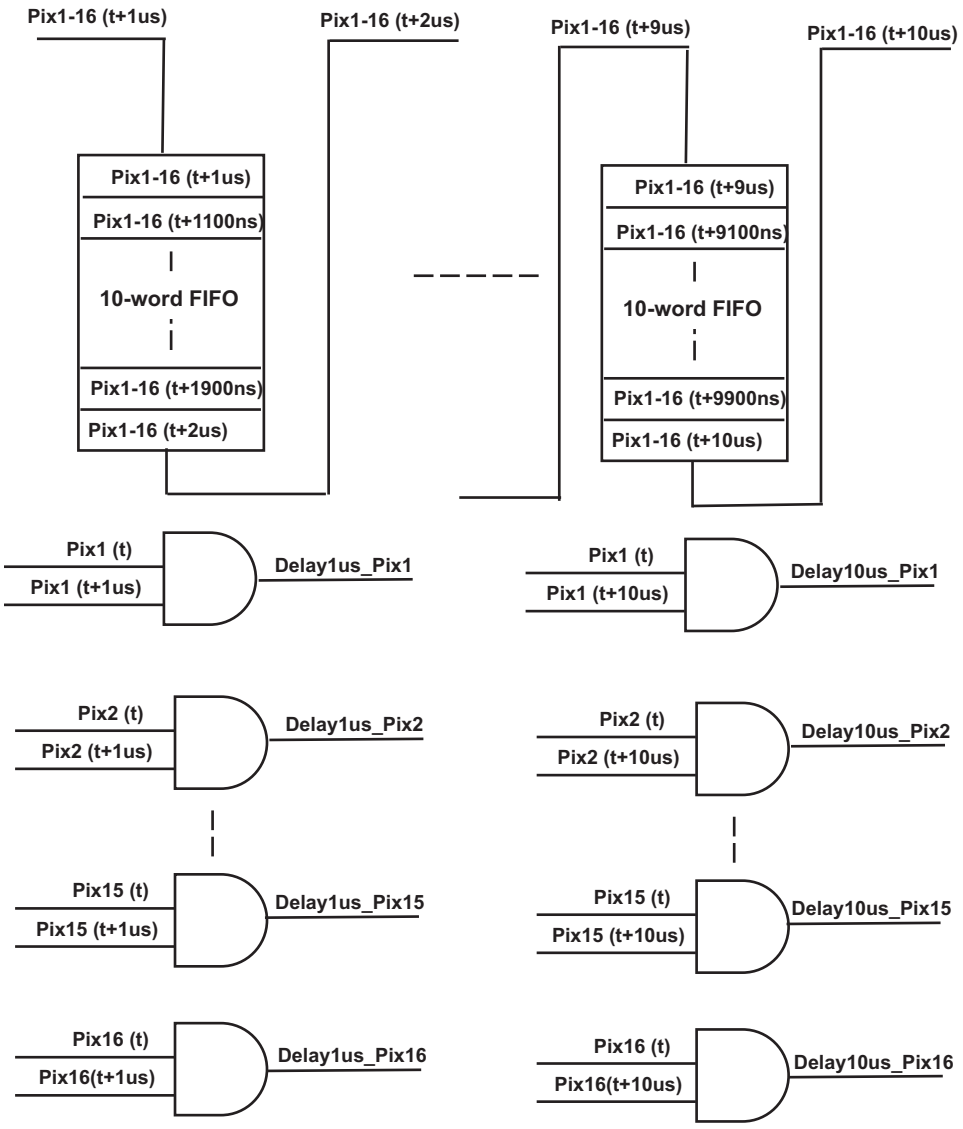

Fig. 10. Level 2 ACF calculation. 10-word FIFOs are used as delay elements to buffer the incoming 100 ns pixel bitstreams and delay them by 1 μs. This allows the ACF to be calculated on the log scale 1–10 μs without sacrificing any of the resolution of the 100 ns input.

functions with different time autocorrelations. The computed autocorrelation data generated from the simulated FPGA PAD was sent back to the testbench interface and stored as data files. These were then compared to the expected ACF for the specific test function.

4.1. Testing procedure

Three different types of input functions were tested

1. A pseudo-random input bitstream (to mimic white noise) which should have a flat autocorrelation function output for all time delays ($j\Delta t$).
2. An input bitstream with a periodic correlation function was emulated using a sine-squared wave probability density function. A bitstream of ones and zeros was generated from this probability density function at a sampling period of 100 ns. Bitstreams of 60 ms duration were generated using several different frequencies of the sine squared wave.
3. An input bitstream with a decaying ACF with characteristic time and decay exponent was emulated by creating a probability density function from a series of Gaussian distributions (Gaussian events) of known variance which were distributed randomly in time. Distributions with different population densities and different Gaussian widths were tested. Bitstreams of sample period 100 ns were created from these parent probability distribution functions over a 60 ms interval for Fig. 13(a and b) and over a 600 ms interval for Fig. 13(c).

5. Results and discussion

Figs. 11–13 compare the results from the FPGA calculation (open circles) and the ACFs calculated in software from the full bitstreams (black curves). Inset in the figures are a portion of the parent probability distribution function from which the input bitstream was created. The software results were calculated for every 100 ns interval between 100 ns and 1 ms. The FPGA PAD outputs 40 data points on a log scale from 100 ns to 1 ms. The FPGA PAD simulated output agrees with the full ACF calculation at every computed point. Also shown in Fig. 13(a–c) is the theoretical ACF for a single Gaussian event. The simulated FPGA PAD output agrees well with these curves, yielding the expected lineshape and characteristic time for each curve.

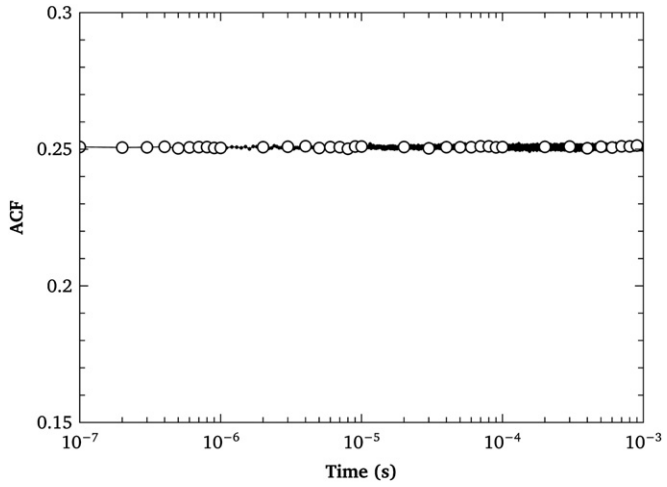

Fig. 11. White Noise ACF—The simulated FPGA PAD ACF output (open circles) is compared with the software-computed ACF output (black curve) from 100 ns to 1 ms for a pseudo-random white noise input, showing the expected flat response.

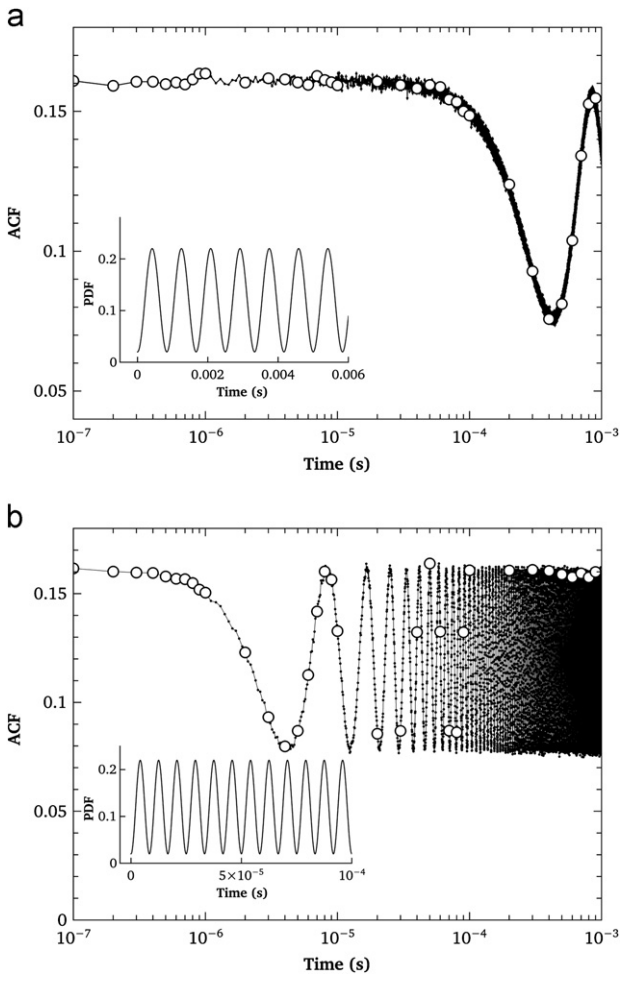

Fig. 12. ACF with periodically distributed input events. The probability density function (inset) follows a sine-squared function and was used to generate an input bitstream. The simulated FPGA PAD ACF output (open circles) is compared to the full ACF computed in software (black curve) from 100 ns to 1 ms on a log scale. The input sequence in (a) has a characteristic period of 0.833 ms (600 Hz sine squared wave) while the sequence in (b) has a period of 8.33 μs (60 kHz sine squared wave).

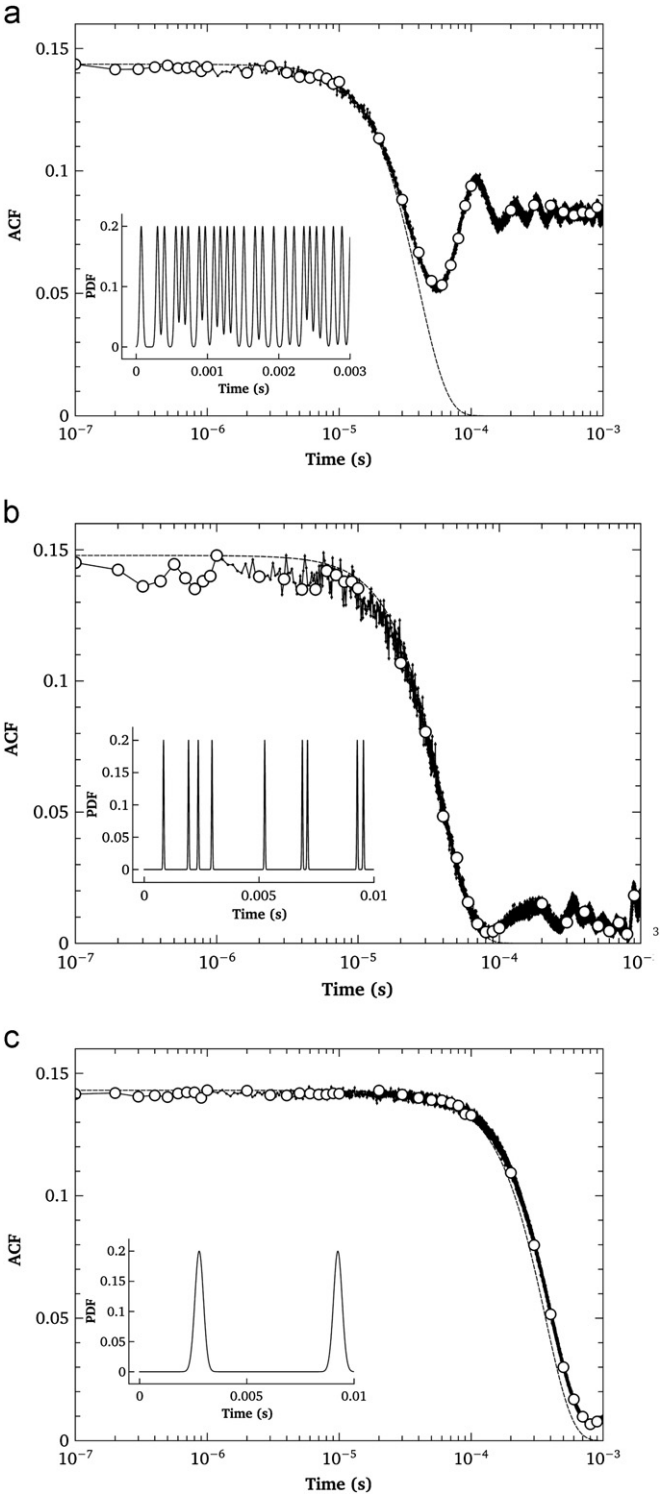

Fig. 13. ACF with characteristic decay time. Probability density functions (insets) were constructed from multiple Gaussian peaks of fixed width which were distributed randomly in time. The ACF output from the simulated FPGA PAD (open circles) from 100 ns to 1 ms is compared to the software-computed ACF computed from the full bitstream (black curve). The theoretical ACF of a single, isolated Gaussian input event (dashed curve) is also shown. Input sequences (a) and (b) are comprised of Gaussian events with $\sigma=20$ μs (characteristic decay time of 33.2 μs) while (c) is comprised of Gaussian events with $\sigma=200$ μs (characteristic decay time of 332 μs). The input sequence in (a) is a more densely populated than that of (b) while the input sequence density in (b) and (c) are the same. The data collection time for (c) was 10 × the duration of (a) and (b).

Note that the calculated ACF curves for Fig. 13(b) show more fluctuations as compared to those seen in Fig. 13(a), which is a result of the lower number of photons in the bitstream used to calculate the ACF in Fig. 13(b). At longer timescales, the ACFs of Fig. 13 have non-zero values due to the non-zero probability of two Gaussian events having a given difference in time. The structure observed at these longer times is due to the limited record length of the bitstreams. As the record length is increased, the long time behavior should approach that of uniform white noise.

In the simulated FPGA implementation shown here, the ACF is calculated at the full 100 ns resolution for 40 points over a 4-decade time scale. Although each of the ACF points is calculated exactly from the bitstream, having the full ACF would, of course, be preferred. This would allow, for instance, a more precise value of the ACF decay constant to be extracted from the data by using curve fitting. Several schemes to optimize the ACF data collection for different classes of experiments could be implemented while still working within the limited resources of the FPGA. Firstly, the distribution of the accumulators within the FPGA can be rearranged in a different configuration so that more data points could be taken around the characteristic decay time of interest. Alternatively, the output data points from the FPGA PAD could be changed from an absolute accumulation of photon correlations for a specific time lag to a running average over a wider time interval. The first option would require several pre-compiled FPGA implementations, possibly one focusing the accumulator resources on each time decade. The averaging alternative does allow for a single FPGA implementation to be used over the course of the experiment but would necessarily incur a trade-off between available FPGA resources and the number of adjacent time lags to be averaged i.e. the precision of the ACF output. If the photon flux is low and if the full 100 ns time resolution is not needed, one could run the ACF accumulators with a slower clock. Naturally, this would average over larger time bins. Note that a slower clock will make the collection of data for the ACF calculation is more efficient for low photon flux data. The implementation of the ACF calculation described here is optimum when the photon occupancy is about 0.5 in a given time bin since the ACF accumulator sum will go as the occupancy squared.

Region of interest ACF calculations on a CCD with small pixels have been achieved on millisecond timescales [14,16] with more recent attempts achieving log-scale, on-chip ACFs at 10 μs timescales [17]. This prototype of the FPGA PAD, however, allows 256 channels of real-time ACF processing down to 100 ns. Additionally, because each frame is not read out individually, the required data transfer rate for this operation is 4 Mb/s as opposed to 2.56 Gb/s required to gain the same information in post-processing from the 16×16 prototype.

The required data transfer rate could be further reduced by placing more of the long timescale ACF processing within the FPGA as described above. Since this implementation uses less than 20% of the FPGA resources, it would be possible to either add level 3 processing or to increase the number of computed ACF delays calculated in real-time. Choosing how to allocate FPGA resources between long or short timescale calculations will depend on the nature of the system under study. This partitioning of resources will need to be optimized for the full scale 128×128 pixel PAD as well.

6. Future work

The FPGA PAD is at an early stage of development. An initial prototype ASIC layer has been fabricated and will soon be tested with an FPGA layer. Both of the clocking modes will be tested as well as the speed of the front-end pixel design. Development of the double-bonded diode layer as well as testing on mated diode-ASIC hybrids of this prototype will follow.

Many challenges remain to develop a full-scale 128×128 tile. Foremost among these are the physical interconnection of the ASIC layer to the FPGA die, tiling a larger area multi-module detector, and communication between the FPGA layers of the different PAD tiles. Modifications to the current design will also be necessary to adapt to a much larger 128×128 pixel array which will use more of the intended Virtex7 FPGA Block RAM and distributed resources.

Inter-communication between FPGAs is particularly important for applications in which calculations are made using readings of more than one pixel. An example of this is the Angular Cross Correlation Function. Higher-order correlation functions and subsequent post-processing have shown the potential to extricate short-range order within disordered systems as exhibited in glassy states [18]. The angular intensity cross correlation function has been used to determine local order including motifs forbidden in periodic crystalline structures [19]. The angular CCF, although similar to the ACF, also has interesting properties that make it more of a challenge to implement on the FPGA PAD. Unlike an ACF where each pixel output is independent, the CCF has spatial dependence and, as such, can tax the intercommunication protocols between chips. Other potential applications of interest to be developed for the FPGA PAD are real-time calibration and real-time image processing algorithms such as the 2D FFT.

Acknowledgments

Supported by DOE Grant DE-FG02-10ER46693 and the Keck Foundation. CHESS is supported by NSF and NIH-NIGMS under NSF Grant DMR-0936384.

References

[1] Z. Huang, K.J. Kim, Physical Review Special Topics Accelerators and Beams 10 (2007) 034801.
[2] E.F. Eikenberry, C. Bronnimann, G. Hulsen, H. Toyokawa, R. Horisberger, B. Schmitt, C. Schulze-Briese, T. Tomizaki, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment 501 (2003) 260.
[3] M.J. Renzi, M.W. Tate, A. Ercan, S.M. Gruner, E. Fontes, C.F. Powell, A.G. MacPhee, S. Narayanan, J. Wang, Y. Yue, Review of Scientific Instruments 73 (2002) 1621.
[4] H.T. Philipp, L.J. Koerner, M.S. Hromalik, M.W. Tate, S.M. Gruner, IEEE Transactions on Nuclear Science NS57 (2010) 3795.
[5] S.M. Gruner, Review of Scientific Instruments 60 (1989) 1545.
[6] M.S. Hromalik, H.T. Philipp, L.J. Koerner, M.W. Tate, S.M. Gruner, Data Acquisition and Control for the LCLS Pixel Array Detector, 2007 IEEE Nuclear Science Symposium Conference Record, 2007, pp. 1744–1750.
[7] L.J. Koerner, M.W. Tate, S.M. Gruner, IEEE Transactions on Nuclear Science NS56 (2009) 2835.
[8] M. Sutton, Comptes Rendus Physique 9 (2008) 657.
[9] P. Falus, L.B. Lurio, S.G.J. Mochrie, Journal of Synchrotron Radiation 13 (2006) 253.
[10] J. Becker, C. Gutt, H. Graafsma, Simulation Study of the Impact of AGIPD Design Choices on X-ray Photon Correlation Spectroscopy Utilizing the Intensity Autocorrelation Technique, Arxiv preprint http://www.elsevier.com/xml/linking-roles/preprint, arXiv:1108.2980, 2011.
[11] W.J. Snoeys, T.A.P. Gutierrez, G. Anelli, IEEE Transactions on Nuclear Science NS49 (2002) 1829.
[12] L. Blanquart, A. Mekkaoui, V. Bonzom, P. Delpierre, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment 395 (1997) 313.
[13] X. Llopart, M. Campbell, R. Dinapoli, D. San Segundo, E. Pernigotti, IEEE Transactions on Nuclear Science NS49 (2002) 2279.
[14] P. Falus, M.A. Borthwick, S.G.J. Mochrie, Review of Scientific Instruments 75 (2004) 4383.
[15] M. Pouchet-Hromalik, G. Seferiadis, N. Huber, M.P. Gough, Review of Scientific Instruments 76 (2005) 094701.

[16] D.R. Schuette, A Mixed Analog and Digital Pixel Array Detector for Synchrotron X-ray Imaging, Doctoral Dissertation, Ithaca, Cornell Universiy, 2008.
[17] J. Buchholz, J.W. Krieger, G. Mocsar, B. Kreith, E. Charbon, G. Vamosi, U. Kebschull, J. Langowski, FPGA implementation of a 32×32 autocorrelator array for analysis of fast image series, Arxiv preprint http://www.elsevier.com/xml/linking-roles/preprint, arXiv:1112.1619, 2011.
[18] P. Wochner, C. Gutt, T. Autenrieth, T. Demmer, V. Bugaev, A.D. Ortiz, A. Duri, F. Zontone, G. Gruebel, H. Dosch, Proceedings of the National Academy of Sciences 106 (2009) 11511.
[19] M. Altarelli, R.P. Kurta, I.A. Vartanyants, Physical Review B General Theory 82 (2010) 104207.