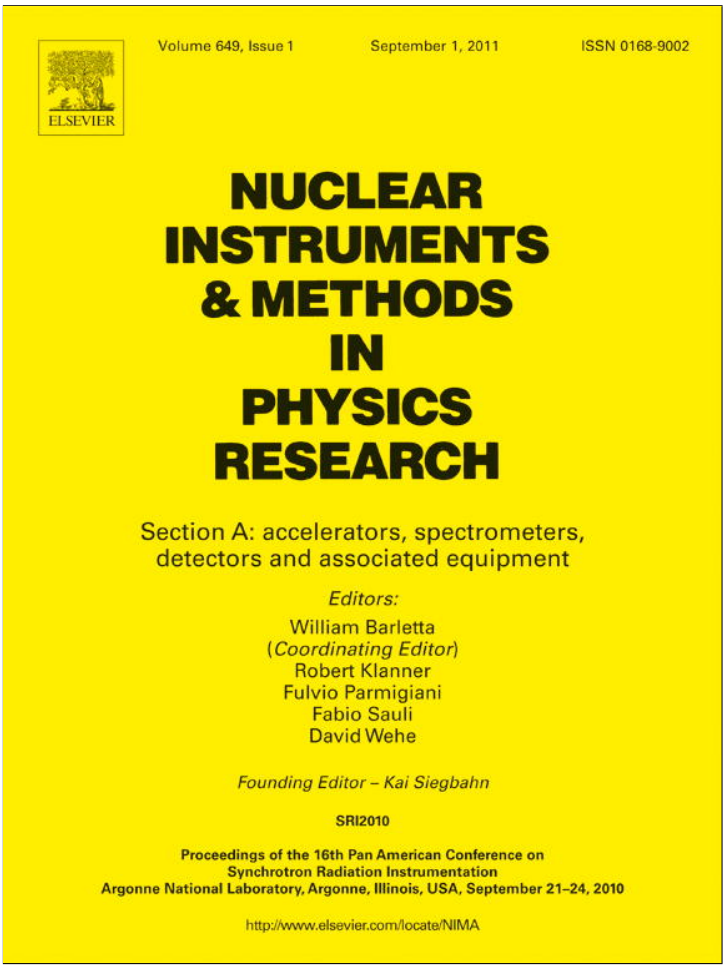
Provided for non-commercial research and education use.
Not for reproduction, distribution or commercial use.



This article appeared in a journal published by Elsevier. The attached copy is furnished to the author for internal non-commercial research and education use, including for instruction at the authors institution and sharing with colleagues.

Other uses, including reproduction and distribution, or selling or licensing copies, or posting to personal, institutional or third party websites are prohibited.

In most cases authors are permitted to post their version of the article (e.g. in Word or Tex form) to their personal website or institutional repository. Authors requiring further information regarding Elsevier's archiving and manuscript policies are encouraged to visit:

http://www.elsevier.com/copyright

Author's personal copy

Nuclear Instruments and Methods in Physics Research A 649 (2011) 237–239

Contents lists available at ScienceDirect

Nuclear Instruments and Methods in Physics Research A

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

Real-time compression of streaming X-ray photon correlation spectroscopy area-detector data

T. Madden *, P. Jemian, S. Narayanan, A.R. Sandy, M. Sikorski, M. Sprung [1], J. Weizeorick

X-Ray Science Division, Argonne National Laboratory, Argonne, IL 60439, USA

ARTICLE INFO

Available online 9 December 2010

Keywords:
X-ray photon correlation spectroscopy (XPCS)
X-ray intensity fluctuation spectroscopy (XIFS)
Field-programmable gate array (FPGA)

ABSTRACT

We present a data acquisition system to perform on-the-fly background subtraction and lower-level discrimination compression of streaming X-ray photon correlation spectroscopy (XPCS) data from a fast charge-coupled device area detector. The system is built using a commercial frame grabber with an on-board field-programmable gate array (FPGA). The system is capable of continuously processing 60 CCD frames per second each consisting of 1024×1024 pixels with up to 64 512 photon hits per frame.

© 2011 Elsevier B.V. All rights reserved.

1. Introduction

In multi-speckle X-ray photon correlation spectroscopy (XPCS) experiments, (partially) coherent hard X-rays illuminating a sample are scattered and create a speckle pattern that is captured by an area detector. Time autocorrelation of fluctuations in the time-resolved speckle patterns yields information on the equilibrium or steady-state dynamics of condensed matter. An optimal area detector for such measurements has a high frame rate so that rapid fluctuations in the speckle pattern are captured but also runs for extended periods so that slower fluctuations can be observed and counting statistics can be improved. Such performance is particularly important when capturing the dynamics in glassy systems where the relevant fluctuation time scales in a particular state can span several decades in delay time from 0.01–100 s [4]. A detector such as this presents a significant operational challenge, however, because if one simply tries to write each data frame to disk then disk space and analysis throughput are rapidly exhausted.

Ideally, the fast frame rate limit is determined by the fluctuation time scale(s) of the sample and could vary from ns or less to more than 1000 s. In practice, however, the fast limit is set by detector technology (and available coherent flux). To be more concrete, we consider a prototypical fast charge-coupled device (CCD) XPCS detector that we use at beamline 8-ID at the Advanced Photon Source (APS), namely the custom-modified SMD detector [2]. It runs at 60 frames per second (fps) with each frame consisting of 1024×1024 2-byte pixels yielding a continuous uncompressed data rate of 126 MB/s. Because each measurement with this detector may run continuously for several minutes the data sets are potentially huge.

A solution to the problem of enormous datasets is to compress the data before it is stored to the disk. To compress data while retaining important scientific information, the compression algorithm must differentiate between signal (photons) and detector noise. This task is greatly simplified by using a detector with extremely high signal to noise ratio such as a direct-detection CCD. For example, with the SMD detector a 7 keV X-ray generates 7,000/3.65 = 1917 electrons in the CCD while the typical read noise is less then 100 electrons so the detector noise is small compared to the signal. Thus, a simple threshold or lower level discriminator (LLD) can be used to compress the speckle data—pixels are stored if their signal is above the LLD and thrown away if below. Very high compression ratios can be achieved because the speckle images are usually very sparse—typically $< 5\%$ of the pixels have signal.

Our first attempt at XPCS data compression used memory buffers on the frame grabber or in the control computer to accumulate as many uncompressed frames as possible (roughly 1000) before pausing acquisition to compress the data and write to disk. This approach allowed us to investigate short-time dynamics but not long-time dynamics. Our second attempt, used software running on the detector control computer to process each image in real time and store the compressed images to disk. Now, we could measure longer-time dynamics but not short-time dynamics because even a powerful multicore workstation could not keep up with the maximum frame rate of the detector. Based on these considerations, we decided to develop specialized firmware to compress the XPCS data before it reaches the internal busses in the computer. The remainder of this paper presents our data acquisition system for real time compression of XPCS data and summarizes its performance.

* Corresponding author. Tel.: +1 630 252 7914.
E-mail address: tmadden@aps.anl.gov (T. Madden).
[1] Current address: Petra-III, DESY, Hamburg, Germany.

0168-9002/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.nima.2010.11.161

Author's personal copy

2. System design and implementation

A schematic overview of our multi-speckle real-time compression system is shown in Fig. 1. It consists of a fast CCD detector (either the SMD camera or the custom-built Fast CCD (FCCD) [1]), a Dalsa Anaconda PCI-X Camera Link frame grabber equipped with Xilinx FPGA, and a host workstation with built-in fast data storage. The software controlling the system is built within the EPICS areaDetector framework [5]. The detector sends frames to the frame grabber at ≤ 126 MB/s via copper cable using the Camera Link standard. As will be described in Section 3, the FPGA compresses the incoming data in real-time. The compressed data is then sent to the CPU at a rate of less than 10 MB/s. The CPU stores the compressed data to disk.

3. Compression method

If we denote a raw image from the detector as $I(i,j)$ where (i,j) denote the row and column coordinates in an image, then an output image suitable for compression, $I_c(i,j)$, is defined as

$$I_c(i,j) = \begin{cases} I(i,j) - \mathrm{LLD}(i,j) & \text{if } I(i,j) > \mathrm{LLD}(i,j) \\ 0 & \text{otherwise} \end{cases}, \tag{1}$$

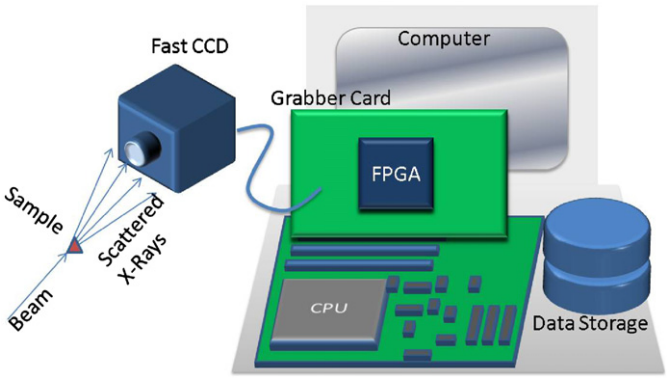


Fig. 1. Architecture of the XPCS data compression system. Data from a fast CCD detector is piped to the FPGA-equipped frame grabber card installed on the detector control computer. The FPGA compresses the data and sends it to the host computer which stores the compressed data to disk.

where $\mathrm{LLD}(i,j)$ is the pixel-specific LLD. When storing to disk, only the non-zero values of $I_c(i,j)$ and the respective pixel coordinates (i,j) are stored in the file.

The LLD is specified on a pixel-specific basis according to the following equation:

$$\mathrm{LLD}(i,j) = \mathrm{DK}_{\mathrm{AVE}}(i,j) + \alpha \mathrm{DK}_{\mathrm{RMS}}(i,j), \tag{2}$$

where (i,j) are pixel indices, $\mathrm{DK}_{\mathrm{AVE}}$ is the pixel-dependent value of the average dark signal, and $\mathrm{DK}_{\mathrm{RMS}}$ is the pixel-dependent value of the root-mean-square (RMS) or standard deviation of the dark signal multiplied by a scalar α where α is typically an integer in the range 1–4. A pixel-specific LLD is required because the mean and RMS dark signal varies on a pixel-by-pixel basis because of X-ray damage to the sensor chip and variation among the ADU's that read out the detector pixels (4 ADU's for the SMD and 96 ADU's for the FCCD).

Because the average dark image is subtracted from the signal frames, the dark averaging is done only before a measurement sequence. But as explained in the caption of Fig. 2, the average dark images need only be acquired and updated periodically, for instance, when the exposure time is changed, because for a given exposure time, the dark mean and variance change very slowly over the lifetime of the detector. The average dark image $\mathrm{DK}_{\mathrm{AVE}}$ is calculated recursively in the FPGA from dark images obtained from the CCD camera.

$$\mathrm{DK}^k_{\mathrm{AVE}} = \left(\frac{\eta-1}{\eta}\right)\mathrm{DK}^{k-1}_{\mathrm{AVE}} + \left(\frac{1}{\eta}\right)\mathrm{DK}^k, \tag{3}$$

where $k \in [1,N]$, $\eta = N$ and $\mathrm{DK}^0_{\mathrm{AVE}}$ is the zero matrix.

Once an estimate of the dark average image is obtained and stored to memory, an estimate of the dark standard deviation is obtained by taking a second set of dark images. The RMS dark image is given by

$$\mathrm{DK}_{\mathrm{RMS}} = \sqrt{\frac{1}{N-1}\sum_{k=1}^{N}(\mathrm{DK}^k - \mathrm{DK}_{\mathrm{AVE}})^2}. \tag{4}$$

Note that calculating the standard deviation requires a square root operation which is difficult to implement in an FPGA. Instead, we estimate the standard deviation by recursively calculating (analogous to above) the mean absolute deviation (MAD) dark image defined as

$$\mathrm{DK}_{\mathrm{MAD}} = \frac{1}{N}\sum_{k=1}^{N}|\mathrm{DK}^k - \mathrm{DK}_{\mathrm{AVE}}|, \tag{5}$$

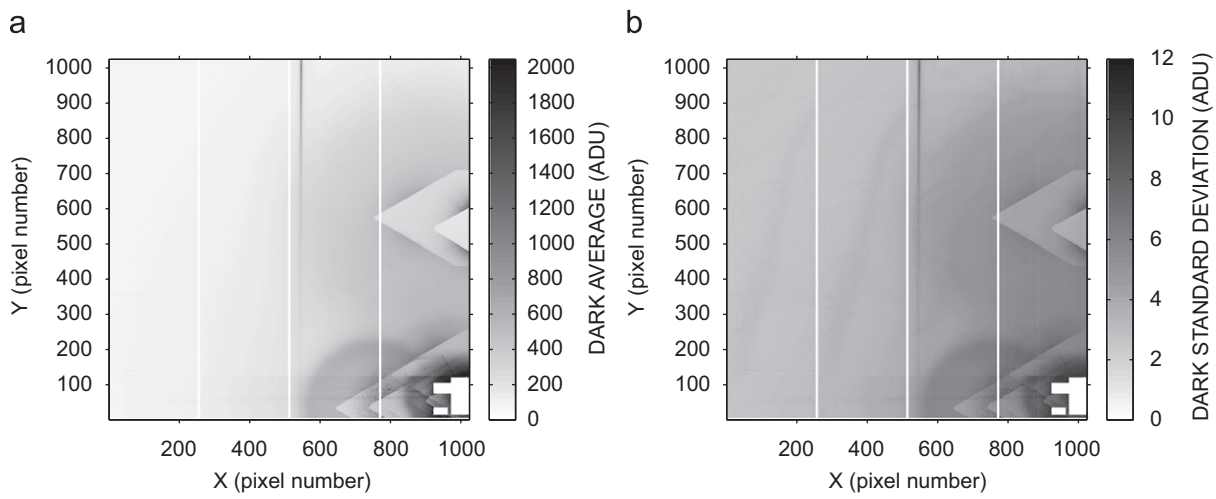


Fig. 2. Typical dark average and standard deviation images for the SMD detector calculated from 512 images acquired at 15 fps. (a) Dark average. (b) Dark standard deviation. X-ray damage to the CCD chip just outside of triangle shapes (right side of images) increases both the dark average and the dark standard deviation. Damage slowly accumulates over months of experiments.

Author's personal copy

and use the relation that for a quantity obeying Gaussian statistics $DK_{RMS} = \sqrt{(\pi/2)}DK_{MAD}$ [3]. We have measured the dark signal produced by our CCD detectors and verified that the noise spectrum is Gaussian. Representative dark average and standard deviation images are shown in Fig. 2.

4. FPGA compilation and software control

The FPGA firmware was written in VHDL ($\approx$10 000 lines of code) and compiled with the Xilinx development tools into a bit file that uses $\approx$45% of the floor area of the Xilinx Virtex II FPGA. Once the frame grabber has been configured using Dalsa's software (Sapera LT), the bit file is loaded into the frame grabber to establish the FPGA configuration. A C++ software driver was written to control the frame grabber and FPGA using the software API supplied by Dalsa. The driver is loaded by the areaDetector [5] EPICS module. Several control screens have been created for controlling the detectors and the FPGA and for directing data storage.

Among many other features, the FPGA control screens allow the user to set the compression rate, the number of images to average to determine the average and MAD dark images, the number of standard deviations to set the pixel-by-pixel LLD (α in Eq. (2)) and the acquisition mode such as pass through, dark subtraction or dark subtraction and compression.

5. System performance

The data acquisition system has been placed into operation at Sector 8-ID at the APS and is performing successfully. The SMD CCD detector running at its maximum speed of 60 fps can stream XPCS data to disk indefinitely allowing the maximum dynamic range of sample dynamics to be studied. We have not yet determined the maximum data rate that the system is capable of processing because we do not have detectors capable of producing data rates greater than 126 MB/s.

There are, however, limitations to our current system. Specifically, the frame grabber requires frames of fixed size, so the FPGA compression works by taking each incoming frame from the camera, making it sparse according to Eq. (1), and then packing P such sparse data frames into a new frame output to the frame grabber. That is, the FPGA compression effectively reduces the frame rate handled by the frame grabber by P. The packing factor, P, is restricted to integer values in the range 1 to 16 so a frame output from the FPGA to the frame grabber contains from 1–16 sparse frames. Because the packing factor must be set before data acquisition begins and the size of the frame output to the frame grabber is fixed, it is possible for the number of pixels exceeding the LLD to exceed the available space in the frames output to the frame grabber. If this happens, information is irretrievably lost but the user is warned of this condition by status updates on the detector control screens. Typically, even if the packing factor is incorrectly set, only a few images out of thousands will be missing pixels. Though the FPGA design could be improved to somehow store these lost pixels for later retrieval, the amount of required engineering is not justified considering the packing factor can simply be set to a smaller value. Specifically, given P and the number of pixels, N_{raw}, in the raw image, the number of compressed pixels N_{comp} per raw frame is limited to

$$\frac{N_{raw}}{4P} - 1024 \le N_{comp} \le \frac{N_{raw}}{2P} - 1024. \quad (6)$$

Because of FPGA technical details, N_{comp} is a range and depends on the actual data in the raw image. For simplicity we assume the worst case (smaller number) in Eq. (6). So, for example, for a 1024×1024 image and a compression factor of 4, we can store 64 512 pixels from each raw image before an overflow condition occurs. Table 1 shows the maximum stored pixels for packing factors 1–16. In practice, however, we have not found this to be a serious issue. First, measured signal rates are usually very low so the maximum packing factor (16) can almost always be used without worrying about overflows. Second, users are encouraged to use a pass-through ("live, no save") mode to preview the scattering before beginning an acquisition sequence so the packing factor can be adjusted, if necessary, based on the observed scattering intensity.

Table 1
Stored pixels versus packing factors.

Packing factor	Max. stored pixels
1	261120
2	130048
3	86357
4	64512
5	51404
6	42666
7	36425
8	31744
9	28103
10	25190
11	22807
12	20821
13	19140
14	17700
15	16452
16	15360

6. Conclusion

We have successfully implemented an FPGA-based system for on-the-fly lower-level discrimination and compression of rapidly streaming multi-speckle XPCS data. The system allows us to acquire and compress 1 megapixel CCD frames at $\le$60 fps allowing XPCS to measure delay times spanning at least 5 orders of magnitude. In conclusion, we note that the compressed raw XPCS data written by this system, is decompressed and autocorrelated via a purpose-built high performance computing (HPC) algorithm, MPICorrelator [7], under the direction of a powerful graphical user interface (GUI) created within the MATLAB framework [6]. Use of the Advanced Photon Source at Argonne National Laboratory was supported by the U. S. Department of Energy, Office of Science, Office of Basic Energy Sciences, under Contract No. DE-AC02-06CH11357.

References

[1] P. Denes, D. Doering, H.A. Padmore, J.-P. Walder, J. Weizeorick, Review of Scientific Instruments 80 (2009) 083302.
[2] P. Falus, M.A. Borthwick, S.G.J. Mochrie, Review of Scientific Instruments 75 (2004) 4383.
[3] R.C. Geary, Biometrika 35 (1935) 310.
[4] X. Lu, S.G.J. Mochrie, S. Narayanan, A.R. Sandy, M. Sprung, Physical Review Letters 100 (2008) 045701.
[5] M. Rivers, areaDetector: EPICS software for area detectors, 2010.
[6] M. Sikorski, Z. Jiang, M. Sprung, S. Narayanan, A.R. Sandy, B. Tieman, A user interface for real-time analysis of XPCS data using HPC, these proceedings, 2010.
[7] B. Tieman, S. Narayanan, A.R. Sandy, M. Sikorski, MPICorrelator: a parallel code for performing time correlations, these proceedings, 2010.