Software-based Integrating Charge-Coupled Devices Signal Processing Using Correlated Double Sampling Technique.

Claudio Chavez Blanco, Carlos Romero Aquino, Jorge Molina, Gustavo Cancelo, Raul Gregor, Jorge Rodas

***Abstract*—This paper discusses a method for reconstructing an image from the digitized output signal of an integrating charge-coupled devices camera (slow scan) based on software by using correlated double sampling and efficient processing resources use for short processing times. The developed software is entirely written in C++ for high speed execution and can work with data acquired by different kind of detection devices with small changes. This work allows to implement a low cost acquiring system for charge coupled devices or provide a development tool suitable to test digital filters, different techniques for pixel computation, or charge-coupled devices signal analysis. Some images produced with real data are shown.**

***Index Terms*—Reading out of charge-coupled devices, correlated double sampling, digital filter.**

I. Introduction

Although charger coupled devices (CCDs) were originally conceived as memory devices, it became immediately clear that the CCD had potential uses that ranged far wider than simple memory applications. Photographic film was the CCD's main competitor as an imaging detector, in the past film has been a tough contender for the CCD due to its performance, cost and ease of use are well established. One historical factor that placed the CCD as leader was the commercial need for an electronic "solid state" imaging detector to replace tube-type detectors. The emphasis was on realizing the CCD's advantages in size, weight, low power consumption, ultra-low noise, linearity, dynamic range, photometric accuracy, broad spectral response, geometric stability, reliability and durability, while attempting to march tube characteristics in format, frame rate, cosmetic quality and cost [1].

Technically speaking, the CCD must perform four primary tasks in generating an image. The first step is called, "charge generation", the second step, "charge collection", that defines the spatial resolution performance for the CCD. The third operation, "charge transfer", that for quantitative scientific measurements, is important to lose as little of the charge as possible during the transferring process. The fourth operation to occur during CCD imaging is the detection and measurement of the charge collected in each pixel. This is accomplished by dumping the charge onto a small capacitor connected to an output MOSFET (Metal-oxide-semiconductor Field-effect transistor) amplifier. This point on the CCD is called the "sense node" or "output diode". The MOSFET amplifier is the only active element that requires power. The CCD array of pixels itself is passive in that it is made up of MOS capacitors. The output amplifier generate a voltage for each pixel proportional to the signal charge transferred [1].

The driving and reading out process of CCDs is well known, there is a lot of commercial hardware available like multi function chips for both task like LM98555 or those options based on embedded system. However, a software to reconstruct the image from a digitized signal either in real-time (processing as signal is digitized) or in a offline mode (signal digitized and stored) will be a valuable tool. Its flexibility will allow to test new techniques for the pixel computation, noise filtering, and it will provide an intermediate level between pure simulation and pure implementation where the real images produced can be analyzed with image processing tools besides analyzing the signal itself. The offline mode is more suitable as development tools because it allows to get a whole image signal just once and work on it the needed time. This paper describes the development of a software for offline CCD signal processing based on digital correlated double sampling technique, but is possible to apply different techniques, i. e. [**?**].

The work is organized as follows: Section II discusses the CCD signal acquisition method used. Section III explains the CDS technique applied to CCD reading out. Section VI discusses the basis of the algorithms used for the software development in C ++, and describes how the images are obtained. Sections V and VI show the results and concluding remarks respectively.

II. CCD video signal acquisition

Fig. 1 shows the current scheme used to collect real data for developing this work. Basically a CCD with a driver that provides BIAS voltages and clock signals. The analog signal from the CCD is amplified by a low noise inverter amplifier, it is passed through an anti-aliasing filter and then to an A/D converter. This converter is controlled by a ML-605 FPGA

Claudio Chavez is with the Laboratory of Mechanics and Energy, Facultad de Ingerniera de Universidad Nacional de Asuncion, San Lorenzo, Paraguay, e-mail: rchavez@ing.una.py.

Carlos Romero Chavez is with the Laboratory of Mechanics and Energy, Facultad de Ingerniera de Universidad Nacional de Asuncion, San Lorenzo, Paraguay, e-mail: cromero@ing.una.py.

Third Name is with the Department of ..., ... Institute of ..., City, Country, e-mail: xxx@xxx.xxx.

evaluation board which also transmit the data to a computer via ethernet protocol in order to save them directly into hard disk. For memory optimization, samples are split into files of 16 MB each. A large image of 4000x4000 pixels can produce more than 8 GB of data depending on sampling rate and ADC resolution.

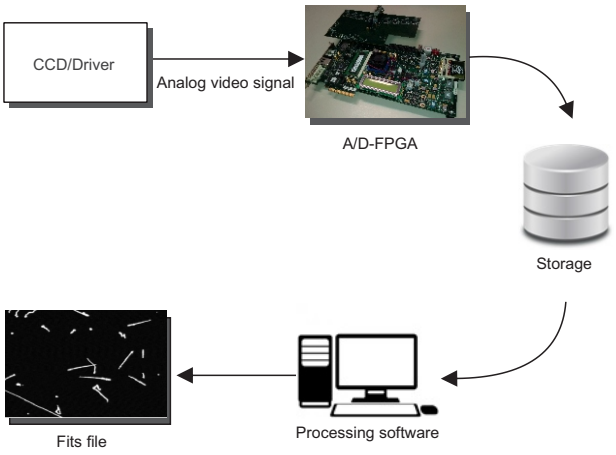

Fig. 1. Signal digitization and storage.

III. Correlated double sampling (CDS) signal

Fig. 2 shows a portion of the signal that represents the information of single pixel in a real data. CCD signal may vary but this zones will be always found.

CDS technique implies to measure a magnitude twice. The first time in a known condition, and the second time in an unknown condition. This technique allows to get rid of certain error introduced by the measuring system (offsets). Digital correlated double sampling (DCDS), the digital version of CDS, is being used more and more in high-performance CCD devices, displacing traditional analog techniques like analog dual slope integration with clamp and sample. The DCDS technique oversamples the CCD output signal to obtain and subtract pixel signal level from pedestal level (reset level). Several samples are used in each zone in order to obtain mean value for further noise elimination [**?**].

To calculate a specific pixel is necessary to obtain the mean reset value and the mean value of the measured signal. The section called reset peak (RP) indicates that a new pixel has started, samples following RP should be integrated (pedestal) until just before summing peak (SP). Samples after SP must be integrated to get the mean signal value. The difference between the average value of pedestal and signal is the pixel value. When more samples are used in integration process (longer integration time) better noise performance will be achieved and because of CDS, undesired offset will be eliminated [2]. The end of line is indicated by a peak much greater than RP.

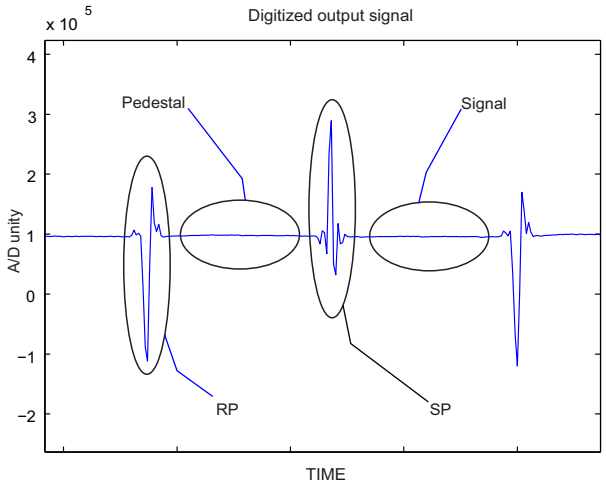

Fig. 2. Digitized output signal of a camera CCD. Circles shows parts that are always found in a CCD signal

IV. Processing software

In this work the software runs in offline mode, which means that samples are already digitized and stored on disk drives. In an online Image extraction all the algorithms are the same, but the part of the software that gets the samples would be different, for example, it may request for samples to an A/D converter. The offline version starts reading the files one by one in chronological order. The name of the files must allow them to be sorted out this way. The first file is read and loaded into memory to start the auto-calibration process. This process is important because it lets the algorithms auto-adapt to variation in signal characteristics, for example, in developing stages of complex reading out systems it is possible that several configuration of input amplifiers are tested. This would modify signal amplitude but not the original shape. Other situation could be the change of sampling frequency, in this case it will not modify anything but the amount of data that represents the same part of the image signal. In both case, if the algorithms rely on fixed parameters to process the signal, they would fail against those variation. For example, setting a fixed amplitude threshold to trigger some functions will produce different results after a change in amplifier gain. The same for the other situation, if a function needs to wait for a fixed amount of time after an event to trigger other functions and uses samples counting for that purpose, there will be errors when sampling frequency changes because that amount of samples represents a different interval of time after the sampling rate variation. The auto-calibration feature makes the software easier to use and very useful in a reading out system designing.

Fig. 3 a very summarized flow chart of steps in auto-calibration process. The key in this process is to set all the thresholds and time intervals (defined as amount of samples) proportional to a measured value of the signal, for example,

maximum peak amplitude and distance between consecutive peaks. When all the parameters needed to extract the image are set and tested, the software goes to a “signal lock” state and it is ready for image extraction functions. Some important parts of the CCD signal are described below:

-Beginning of Image (BOI): Indicates when a image starts.

-End of Line peak (EOLP): it marks the end of a row in the image.

-Find the minimum Reset Peak (RP): it marks the beginning of a pixel.

-Pixel Guard (PG): It represents the amount o samples which must be discard from pixel calculation process. They are part of RP and EOLP transients and it can introduce noise in the image.

-Integration Width (IW): It represents the amount of samples available for pixel value calculation.

Fig. 4 and 5 show the parts described above.

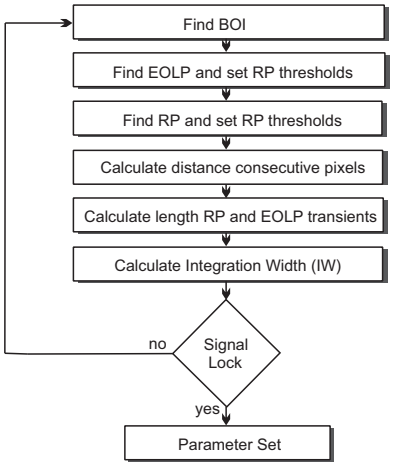

Fig. 3. Auto-calibration process.

Once the parameters mentioned are set, the next line is processed. The whole process of extracting an Image can be summarized in the following steps:

1) Read file(n).
2) Find Pixel (RP) peaks.
3) Compute pixels for each line.
4) Save image matrix into a fits file.

Usually several image rows are saved into the same file, depending on the resolution of the image, sampling rate and A/D converter resolution, between 2 and 10 complete rows can be placed in a 16 MB file. Those files were given with a clock-date based name in order to recognize how they were taken in time. The program uses those names to set the correct sequence for reading the files. It is important to remark that because of splitting, there will be rows whose samples are split in consecutive files. The software must recognize those rows in order to process them as a single row. These process are performed in the first step.

In the Second step, the program looks for RPs. This is done by selecting samples which satisfy two conditions:

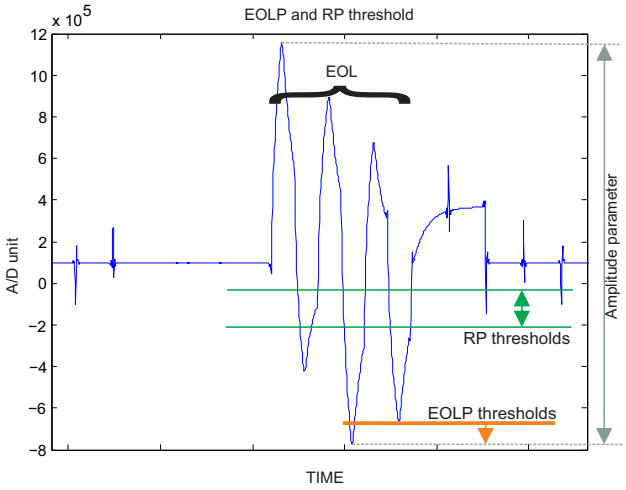

Fig. 4. EOLP and RP threshold.

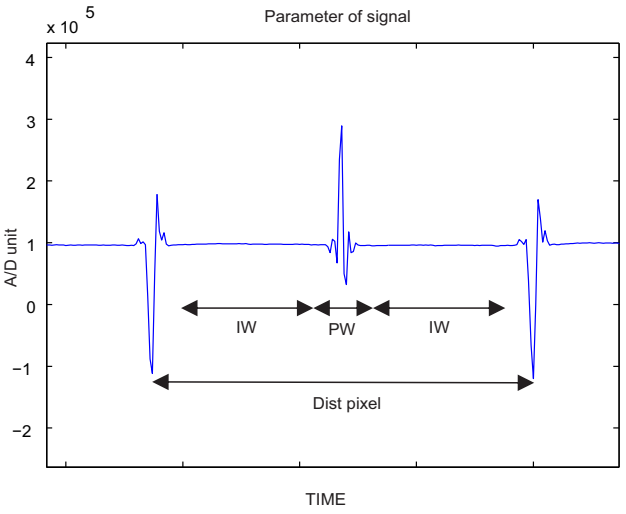

Fig. 5. Time parameters.

1) The peak condition:

$$Sample(k-1) > Sample(k) < Sample(k+1) \quad (1)$$

2) And the threshold condition:

$$Sample(k) < RP\_threshold \quad (2)$$

The position of samples which meet those condition (RP) are saved in a buffer. Those positions need to be checked by a correction filter in order to discard false RP detection and/or fill in a missed RP. These error can be triggered by saturated pixels which deform the signal around them. It is important to note that any misdetected or missing pixel will

produce a misalignment row that will corrupt the image. The Algorithm developed for correction of RP misdetections uses the known distance between pixels to check if the separation of consecutive pixels are correct. If the distance is smaller or bigger than the known distance, a discarding or filling process must be performed. The RP detection runs until an EOLP is found, then the next step is executed.

Third step performs pixel computation using the formula:

$$Pixel = Average(Signal\,value) - Average(Pedestal\,value)$$

The fourth step represents the loop of steps second and third. For each line, the RP searching process is executed until an EOL is found. Then, pixel computation is launch with step third, and then the program goes back to second step. Each computed row is saved into an image matrix in memory and this process repeats until the end of image.

Finally, the image matrix is saved in Flexible Image Transport System (FITS) format. This process will depends on what language is used. In this case the open source library CFITSIO of NASA was used.

Fig. 6 shows a scheme of the algorithms above described. As noted, just one file is loaded into memory and joined with the incomplete last line from previous file. All the processes are performed reading this large vector in memory, which is dynamically reserved according to the size of file. This allows the software to work with less than 70 MB of RAM for images of 4000x4000 pixels. Memory usage can be decreased by storing every line directly into disk as soon as they are computed, but this would increase the processing time.

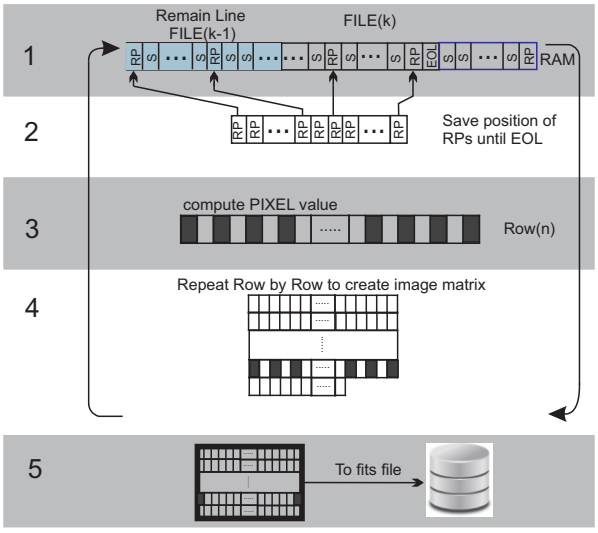

Fig. 6. Time-flow scheme. Each

V. Results

Several images were reconstructed from stored samples. Some signals differs in shape due to use of a filter embedded in A/D converters that produces transient signal (ringing) in reset and summing peaks. Two different A/D converters were used, indeed, the software was able to auto-adapt to both kind of signals and extract the images. Samples used in this work were taken from a CCD testing stand located at a lab at FERMILAB. Signals were digitized using an XXXXX A/D converter controlled by a ML-605 Virtex-6 based FPGA development board and transferred through an ethernet connection to be stored directly into disk drives with chronological named files. For the first type of signal, Fig. 7 shows part of an image extracted from the first type of signal, with 1250x549 resolution pixels, the whole process lasted a average computation time between 3 and 6 seconds in normal home computers.

Fig. 8 shows part of an image from a second type of signals, this time a larger image of 4000x4000 pixels, and the average time to extract it was between 25 and 35 seconds in a normal computer (AMD Phoenom® II X4 965 processor with 4 GB of RAM and Open SUSE 14).

These results show that is in fact possible to perform an online extraction of the images. In this kind of *slow scan* CCD cameras, where usually the exposure time is measure in minutes or even hour, and the reading out time is also considerable due to large integration times (1-5 minutes for 16 Mpix CCDs), the samples can be processed as soon as they come out from the A/D converter because the processing time is less than exposure and reading out time. In these test most the time used to extract the images was lost in disk accessing and joining split rows.

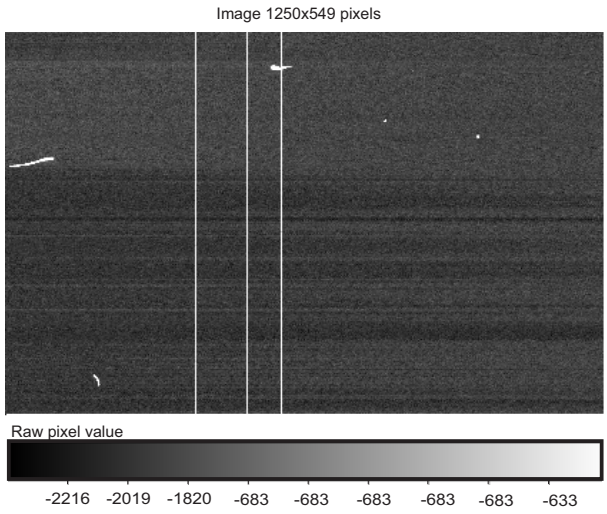

Fig. 7. Image 1250x549 pixels, data type float.

VI. Conclusion

Software-based CCD signal reading out is proved to be very useful option for development stages of more complex hardware/software system. As shown, it is possible to perform a complete reconstruction of image easy and efficiently from digitized slow-scan CCD signals by using only software processing. Although a single large image needs several GB, just few images can be digitized and then use software processing to test pixel computation techniques, digital filtering or equalization. The flexibility of software-based method allows to test

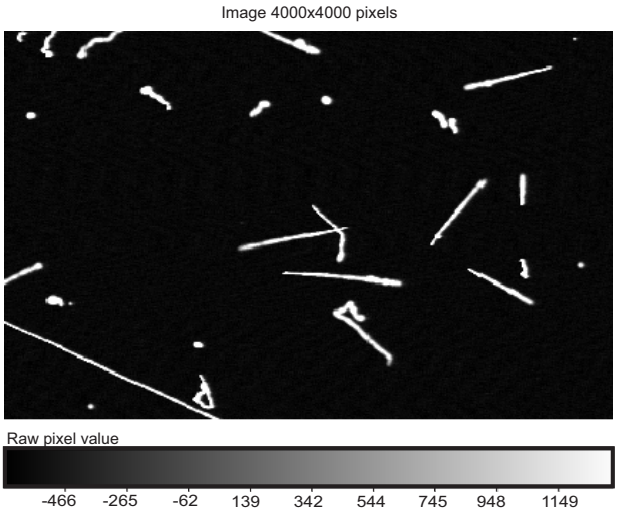

Fig. 8. Image 4000x4000 pixels, data type float.

these options without hardware changing and compare results with the same images using image analysis tools. The code can be used in embedded systems with small changes, or it can be optimized to run as a stand-alone application on a custom soft-processor synthesized in a FPGA (like Microblaze®) for online fast image extraction. This might be needed to read out images from several slow scan CCDs at the same time without storing samples.

REFERENCES

[1] J. R. Janesick, *Scientific charge-coupled devices*. SPIE press Bellingham, Washington, 2001, vol. 117.

[2] R. Liu, J. Lu, J. Ding, and Y. Liu, "Ccd signal processing based on correlated double sampling," in *Bioinformatics and Biomedical Engineering, (iCBBE) 2011 5th International Conference on*, May 2011, pp. 1–3.