Distributed continuous event – based data acquisition using FlexRIO FPGA

C. Taliercio, G. Manduchi, A. Luchetta, A. Rigoni

***Abstract*–High-speed event driven acquisition is normally performed by ADC boards with a given number of pre and post trigger samples that are recorded upon the occurrence of a hardware trigger. A direct physical connection is therefore required between the source of event (trigger) and the ADC because any other software-based communication method would introduce a delay in triggering that would turn out to be not acceptable in many cases.**

This paper proposes a solution for the relaxation of the event communication time that can be in this case carried out by software messaging (e.g. via a LAN) provided that the system components are synchronized in time (e.g. using IEEE 1588 synchronization mechanism). The information about the exact event occurrence time is contained in the software packet sent to communicate the event and is used by the ADC FPGA to identify the exact sample in the ADC sample queue. The length of the ADC sample queue will depend on the maximum delay in software event message communication time.

A prototype implementation using a National FlexRIO FPGA board connected with an ADC device is presented as proof of concept.

I. Introduction

Analog to digital converters (ADCs) are extensively used in fusion experiments to collect a variety of signals such as signals from electromagnetic probes and plasma diagnostics. In the past, for plasma discharges of short duration, transient recorders have been normally used. Once initialized, a transient recorder waits for a trigger signal continuously filling a circular buffer. When the trigger has been received and the programmed number of post trigger samples acquired, the transient recorder stops and the content is read afterwards. Knowing the trigger time and the exact clock frequency it is possible to tag each sample with time.

The data acquisition paradigm has changed in the recent experiments running long lasting plasma discharges. In this case continuous data acquisition is required because the amount of the required memory to store acquired signals before readout would be too large, and in any case it would be not acceptable to wait until the end of the pulse before being able to look at signals. Handling a stream of data from the ADC to computer memory, and eventually on disk, limits the maximum sampling speed that can be achieved in acquisition. Appropriate buffering techniques can be adopted to increase data throughput but in practice no more than some tens of Msamples/s can be handled in ADC data streaming, to be divided by the number of ADC channels. Larger sampling speeds do not normally permit continuous data streaming as they would produce an unmanageable amount of data. High speed data acquisition is for this reason usually triggered by events, either synchronous or asynchronous, signaling the occurrence of some physical phenomenon that requires a larger dynamics in data acquisition during a time window centered on the event time. Commercial solutions exist (see e.g. [1]) for this kind of data acquisition combining the use of a circular buffer with continuous acquisition. When a trigger signaling the event occurrence has been received and the programmed number of post trigger acquired, the pre and post samples describing the evolution of the signal during the specified time window around the event are transferred to computer memory while the ADC is again recirculating the input buffer waiting for the next event. In this way it is possible to complement continuous data acquisition at relatively low frequency with high speed data bursts acquired on event occurrence, using the combination of two different hardware solutions.

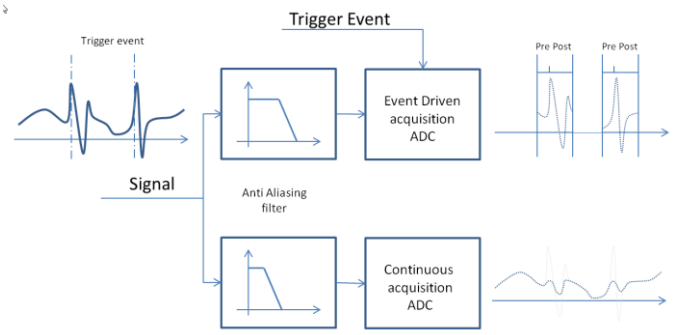

Fig. 1 Continuous and Event Driven Acquisition. Implemented with different hardware solutions.

In order to provide a correct association of samples with time it is necessary to know the precise time of the trigger and the exact frequency of the conversion clock provided to the ADC device. The usual approach adopted in fusion experiments is the use of a timing system that provides phase locked clock signals and triggers at a (normally programmable) set of times. A link, normally in fiber optic, connects all the timing devices in the experiment plant and brings a synchronizing clock signal, possibly encoding timing events. Every connected timing device will derive phase locked clock signals and triggers feeding the local ADC devices.

Manuscript received April 3, 2016. (Write the date on which you submitted your paper for review.)

The Authors are with Consorzio RFX (CNR, ENEA, INFN, Università di Padova, Acciaierie Venete SpA), corso Stati Uniti, 4 – 35127 Padova, Italy (telephone: +39-049-8295039, e-mail: gabriele.manduchi@igi.cnr.it).

A different approach in timing synchronization has been introduced by the IEEE1588 Precise Time Protocol (PTP) [2]. Using IEEE1588 PTP the network used for computer communication is also used to synchronize the computer clock by means of a message passing protocol synchronizing a set of slave devices to the reference clock. The precision in time that can be achieved in synchronization depends on the quality of the network communication and in particular on the jitter in communication times. A precision of 100 ns in synchronization has been measured in tests performed with PTP [3] and it can be further reduced with a properly configured network, using PTP aware routers. This precision is enough for most data acquisition requirements in fusion devices and PTP is foreseen in ITER, where every component involved in data acquisition and control will be synchronized to a grand master clock using GPS for maximum time precision. PTP can be used both to synchronize CPU clocks to achieve time precision in software applications and in dedicated hardware devices (see e.g. [4]) to generate a phase locked clock signal and trigger signals at pre-programmed times. In this way there is no need for a dedicated physical link connecting the timing devices. This provides a great simplification cabling especially in large experimental plants. There is however still the need for a physical connection in event triggered data acquisition between the event (trigger) source and the target ADC devices. In this case, in fact, it is not possible to select in advance the time of the trigger in PTP aware timing devices and a direct physical connection is required, leading again to a cabling configuration not dissimilar from that of the former timing systems. Sending the trigger information via the network is not acceptable due to unpredictable delay in transmission that would not allow a proper reconstruction in time of the acquired samples.

In this paper a different approach is proposed for handling data acquisition triggered by asynchronous events, based on the following two assumptions:

- Every component of the system has a precise cognition of time;
- There is a data link between event detectors and target ADC devices.

Observe that the second assumption must be satisfied if both the event generator and the ADC device are synchronized via PTP. Flexible distribution of events such as multicast can be achieved in this way via software.

Using a data link for event communication introduces some unpredictable delay between the event generation time and its reception time. For this reason, a timestamp bringing the exact time of the event occurrence will be associated with the event data packet. The receiving ADC will hold recent signal history in a circular buffer, providing at the same time a subsampled version of the (filtered) input signal for data streaming. Based on the notion of time and on the time tagging the trigger data packet it is also possible to provide, in a separate output channel, the set of samples acquired at full conversion speed. The internal circular buffer will be wide enough to keep the signal history for a time equal to the sum of time required for event transmission and the desired time before the event occurrence in the time window for acquisition.

The management of the circular buffer as well as the proper tagging of acquired samples with time will be carried out by the FPGA supervising ADC operation. In the paper a prototype implementation based on NI Flex RIO [5] is presented as a proof of concept. Using FlexRIO for prototyping FPGA applications offers several advantages such as short development time, if compared with direct HDL programming, and may represent also a solution for production where a small number of ADC channels is involved. For more channels, this solution could be unacceptably expensive. For this reason, it is foreseen that the developed algorithms will be implemented in low-cost FPGA-based boards.

II. Prototype Development

A proof of concept system has been developed using the following National Instruments devices:

- NI PXIe 7962R: the FlexRIO device hosting the FPGA;
- NI5751: ADC adapter for FlexRIO, 16 Channels 50 MSamples/s, 14 bit. The acquired samples are directly available in the FlexRIO FPGA;
- PXI 1073: rack hosting the FlexRIO and a bridge for MXI communication with a NI PCIe 8361 board mounted in the host PC.

Windows is running in the host PC because LabVIEW FPGA Module for FPGA development for cRIO and FlexRIO is supported only on Windows OS. In this initial version of the prototype PTP is not used for time synchronization. In order to simulate time synchronization, a timing module is used to generate both a clock signal for ADC conversion and a pre-programmed trigger signal to be sent to a waveform generator that produces the waveform to be acquired. Shortly after the trigger generation, the PC controlling the timing module will send a data packet to the FPGA bringing the trigger time, thus emulating the delayed reception of the trigger message.

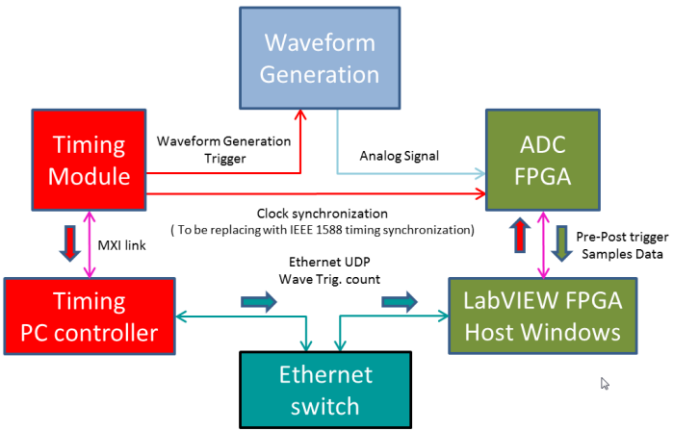

Fig. 2 Hardware and data flow schema implemented in the proof of concept test.

When the generated waveform is a rising edge generated by the waveform generator upon the occurrence of the trigger,

the proper behavior of the system can be checked by visualizing the acquired samples sent by the FPGA program in correspondence to the pre-programmed window around the event time. If everything works correctly, the edge will appear after a number of samples equal to the programmed number of pre-trigger samples. In this application, acquired samples are sent via communication FIFO to LabVIEW for visualization. In a more realistic environment not involving LabVIEW, samples fill be sent via the communication FIFO to routines running on the host system (Windows or Linux) and that will eventually store data on disk. Two different outputs FIFO are defined in the system: one for streaming subsampled data, and the other for high frequency data bursts upon event occurrence.

The FPGA program, developed in LabVIEW, is organized in three different single cycle timed loops. The first loop runs at 50 MHz, that is the sampling frequency of the ADC. In this loop the input samples from the ADC are stored in one internal FIFO able to store up to 4096 samples per channel. The same channels, after filtering and subsampling are sent to the first output FIFO to provide continuous data streaming.

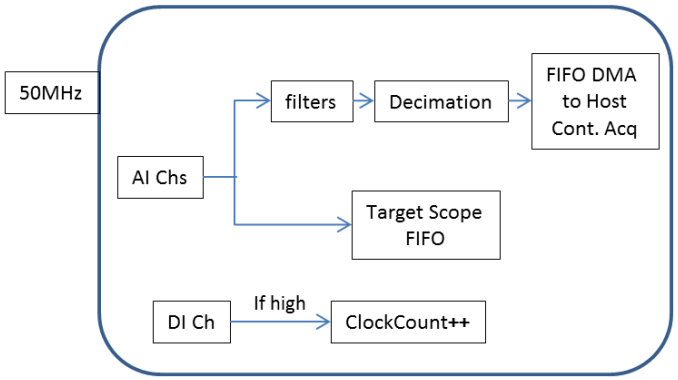

Fig. 3 FPGA loop schema acquiring analog signals, filtering and clock synchronization

The second loop runs at 100 MHz and supervises the management of a circular buffer stored in DRAM. This is the buffer holding the recent history of acquired signals and it will be used for retrieving samples corresponding to the time window around the event time. If a trigger message tagged with time T1 is received at time T2, the sample corresponding to the actual trigger generation will be stored in the DRAM at an address corresponding to (T2-T1)*F previous samples in the circular buffer, where F=50E6 is the sampling frequency.

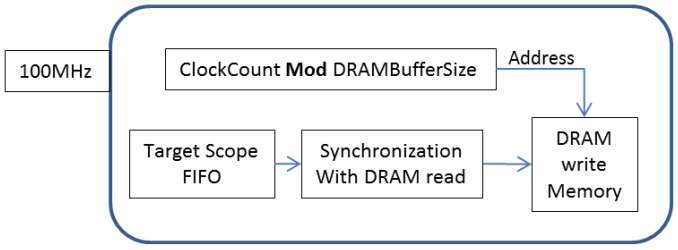

Fig. 4 FPGA loop schema writing DRAM

The use of DRAM for the circular buffer is mandated by the fact that a large number of samples must be maintained in the history to account for realistic delays in trigger message transmission. Here, a 1 MSample DRAM per channel is used, making it possible to handle a delay in trigger notification up to 20 ms. The 4096 samples internal FIFO (per channel) is necessary to handle DRAM memory access. At every cycle the DRAM state is checked and, if the DRAM is available for store, a new sample is transferred from the FIFO, if present. The cycle time of this loop is shorter than that of the first loop so that it is possible to consume the internal FIFO after the DRAM pipeline is fully operational.

The third loop, freely running, monitors trigger input. When a trigger with associated time is detected, the indexes on the DRAM circular buffer are computed based on the current time maintained in step by providing a synchronized clock to the FPGA and the ADC. The trigger time, and DRAM readout is initiated, reading a sample from the DRAM, if available, at every cycle.

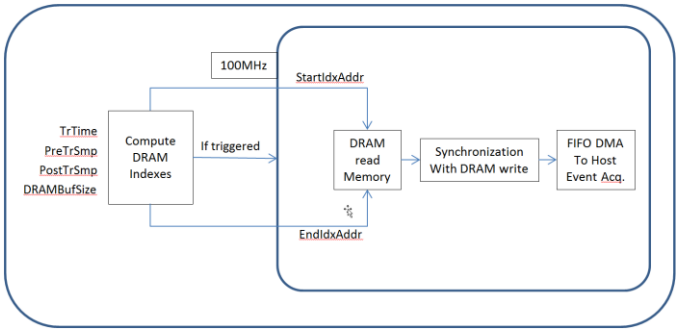

Fig. 5 FPGA loop schema monitoring Event Trigger and communicating data to Host via DMA

A state machine supervises this loop and when in DRAM reading state, at every cycle the status of the DRAM is checked for the availability of a new read sample and the output FIFO is checked for its availability in sending a sample. When both conditions are met, the sample is transferred from the DRAM to the output FIFO. When a number of samples corresponding to the required samples in the time window have been transferred, the system is ready to receive a new trigger notification. In order to avoid too much interference in DRAM write access, possibly overflowing the internal FIFO (the used DRAM does not support a dual port), read requests in the third loop are interleaved with write requests in the first loop using a shared register. The interleaving logic is summarized in blocks "Synchronization with DRAM Read" and "Synchronization with DRAM Write" in figures 4 and 5, respectively.

III. Future Work

The proof of concept has been developed based on the supervision of a Windows PC running LabVIEW. When the system is used in production it is however unlikely that LabVIEW will be used and therefore we investigated the feasibility of two possible evolutions of the system.

The first approach retains the usage of NI hardware and in particular of the FlexRIO as FPGA, but uses Linux-based data acquisition. In this case Windows LabVIEW would be replaced by the NI-RIO driver for Linux, providing the run-time support for RIO devices, that is, the management of the Input/Output from/to Linux software and the FPGA. Input for setting pre and post samples and or configuring a set of parameters and output for reading the continuous and burst data streams will be managed by programs using the NI-RIO drives and communicating with the data system for signal storage on disk. In this configuration, PTP synchronization can be carried out by a PTP aware timing devices for:

- Tagging event generation with precise (absolute) time, that will be included in the trigger message;
- Providing a synchronized clock to the FPGA and the ADC so that times internally computed from clock counters can be trusted.

This configuration is being considered for the ITER Neutral Beam Injector (NBI) Test facility currently under construction at Consorzio RFX, Padova (Italy)[6]. Currently a traditional timing system is used for the first of the two scheduled experiments, but the use of IEEE 1588 PTP is foreseen for the development of the ITER size NBI [7]. The foreseen number of signals requiring this kind of acquisition is small thus justifying the cost per channel of this solution (considering the investment in development time for the alternative solution listed below).

In the longer term, we are considering investigating a different possible approach, that is, porting the algorithms implemented in FlexRIO under System On Chip (SoC) boards,such as Red Pytaya[8], Parallella [9] or Zedboard [10], all using the Zynq solution. The SoC architecture provides a tight connection between the FPGA and the processor and therefore it represents the optimal solution for this kind of application, where the CPU supervises communication over data link and interaction with the data system and the FPGA carries out the required buffering and trigger management.

REFERENCES

[1] CAEN DT 5720 Available: http://www.caen.it/csite/CaenProd.jsp?parent=14&idmod=624

[2] IEEE 15800 standard Available: https://standards.ieee.org/findstds/standard/1588-2008.html.

[3] A. Soppelsa, A. Luchetta, G. Nmanduchi "Assessment of Precise Time Protocol", *IEEE Trans. Nucl. Sci.*, Vol: 57, pp: 503-509, Apr 2010

[4] NI PXI 6683 Available: http://sine.ni.com/nips/cds/view/p/lang/it/nid/211064

[5] NI FlexRIO Available: http://www.ni.com/flexrio/#

[6] P. Sonato, P. Agostinetti, G. Anaclerio, V. Antoni, O. Barana, M. Bigi, *et al.*, "The ITER full size plasma source device design", *Fusion Engineering and Design*, vol. 84, no. 2–6, pp. 269-274, Jun. 2009

[7] A. Luchetta, G. Manduchi, C. Taliercio "Current status of SPIDER CODAS and its evolution towards the ITER compliant NBI CODAS", this conference.

[8] Red Pitaya Available: http://redpitaya.com/

[9] Parallella Available: https://www.parallella.org/

[10] Zedboard Available: http://zedboard.org/