Ethernet Network-based DAQ and Smart Sensors for the OPERA Long-baseline Neutrino Experiment

C. Girerd[1], S. Gardien[1], J. Burch[2], S. Katsanevas[1], J. Marteau[1]
[1]Institut de Physique Nucléaire de Lyon, CNRS-IN2P3 et Université Claude Bernard Lyon I
69622 Villeurbanne Cedex
[2]Agilent Laboratories, Wireless Solutions Department
Palo Alto, CA 94304

Abstract

We propose a data acquisition scheme for the OPERA long-baseline neutrino experiment based exclusively on Ethernet. The expected data rate allows the use of an Ethernet capable device close to the sensor. The basic idea is to build a distributed acquisition system on Ethernet, made of about 1000 nodes of 64 channel front-end modules. Each node will be controlled and readout by an embedded Ethernet chip and will be therefore transparently visible on the network. These modules will be directly controlled by the event building computers and will be able to execute a set of embedded functions for slow control monitoring and data readout. We present a first prototype of Ethernet capable front-end module with high speed ADC and TDC capabilities. A custom ASIC from Agilent Laboratories has been used. This device includes an embedded web server and implements the IEEE 1451.2 standard. This paper presents a full hardware implementation of a Smart Interface Transducer Module (STIM) defined by the IEEE 1451.2 standard. We describe a STIM architecture dedicated to the front-end control and readout, designed in VHDL and synthesized in a FPGA. We present preliminary results and functional validation of the control and readout functions through Ethernet and simple web browser tools.

I. Introduction

The OPERA long baseline neutrino experiment has been approved in September 2000 and aims to test the hypothesis of neutrino oscillations by detecting the appearance of neutrino-tau events in a pure neutrino-mu beam [1]. The neutrino beam will be produced at CERN by mid-2005 and will be targeted on the OPERA detector located 720 km away in the Gran-Sasso underground laboratory (Italy).

The detector (Figure 1) will be composed of 3 identical super-modules. Each super-module will include 2x24 (X, Y) scintillator planes interleaved with walls of emulsion bricks. Each projection plane, will consist of 256 scintillator strips readout by a Wave Length Shifting (WLS) fiber. The emitted light will be collected by a multi-pixel device, a 64 Multi-anode Photo-Multiplier Tube (MaPMT) or by a multipixel Hybrid Photo-Diode (HPD) with the same number of pixels. An auto-triggerable 64 channels front-end chip will be used for charge pre-amplification and signal shaping. Each channel contains a slow shaper (peak time 1 μs) for the charge measurement and a fast shaper (peaking time 75 ns) for the trigger generation. The logic OR of the 64 trigger outputs is used to form a common trigger signal to all channels. Each X-Y plane will be thus readout by 16 front-end chips. The event time stamping requires a synchronization mechanism between each module with an accuracy of about 10 ns. The aim of the tracker is to locate the emulsion bricks likely to contain an interesting event in order to retrieve these bricks for a post-processing. Of the order of 20.000 beam related Charge Current events are expected in 5 years. The event trigger rate is therefore dominated by the electronic noise and the photo-detector dark current. The peak rate does not change substantially this estimation. An expected trigger rate of few 10 Hz/channel is a reasonable estimate of this dark current and it is currently under evaluation. This relatively low data rate allows to use Ethernet to directly connect the front-end detectors to the event building CPU's.

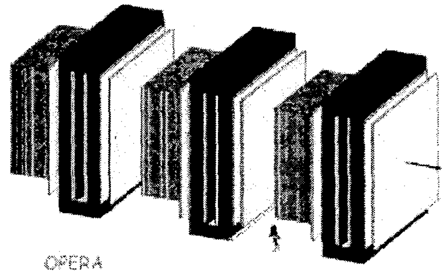

Figure 1: OPERA detector overview.

Nowadays, Ethernet is widely used in most Data Acquisition System of High Energy Physic experiment. High speed products such as Gigabits switches are available at low cost and 10 Gigabits products are now emerging. Full duplex capability allows to build very high speed and reliable point to point links at a reasonable cost. In general, for high data rate experiments, Ethernet is still limited to the higher levels of the DAQ. The use of specific links bus and protocols is still preferred for the front-end electronic readout. However, for lower bandwidth experiments the use of Ethernet down to the sensors as a unique network for data collection and slow control is now conceivable at a reasonable cost.

We propose such a distributed data acquisition system for the OPERA experiment. It will be based exclusively on smart sensors directly connected to Ethernet. The main idea is to design a 64 channel Ethernet Capable Front End Module. Each module is autonomous and possesses its own Internet Protocol (IP) address and embeds all the necessary functions for slow control, monitoring, and readout of the front-end analog chip.

0-7803-6503-8/01/$10.00 2001 IEEE

II. ETHERNET BASED ACQUISITION SCHEME FOR THE OPERA EXPERIMENT

The overall acquisition scheme is shown in figure 2. The global architecture is based on three levels. The first level contains 1152 nodes of 64 channels Ethernet capable front-end module directly connected to Ethernet and close to the detector. The second level is made of 10/100 Mbps switches that concentrate the front-end module outputs of each X-Y plane (72 in total). The last level is a gigabit switch that allows the connection to the event building PCs and storage devices. Note that an additional PC could be easily connected to each X-Y plane if a local processing is needed such as for instance X,Y coincidences.

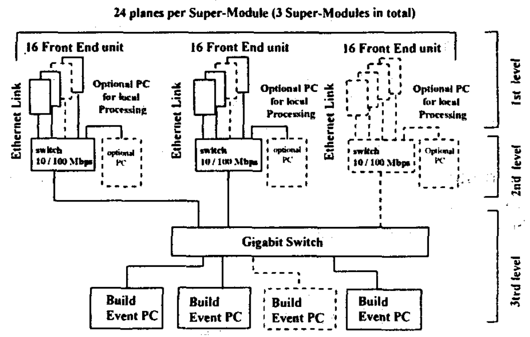

Figure 2: Network based acquisition system architecture.

The advantage of such an approach is the possibility to use standard, reliable and low cost products (i.e. Ethernet cables, PC, switches). The system can be easily upgraded and scaled up if more processing power is needed.

Table 1 gives the expected load at different levels of the network assuming a very conservative trigger rate of 100 Hz per channel and a digitized hit size of 64 bits (charge, address and time-stamp).

Table 1
Expected network loads at different level

	Nominal Load / link	Maximum Capacitance/ level
1^{er} level	500 Kbps	10 Mbps
2^{nd} level	8 Mbps	100 Mbps
3^{thrd} level	192 Mbps	1 Gbps

A single Gigabits switch for each super-modules provides a sufficient security factor. The processor loads and the network modelization are currently further evaluated. In the following sections of this paper we will focus on the network aspects at the front-end side and principally on the Ethernet capable front-end module design.

III. NETWORK CAPABLE FRONT-END MODULE

The final goal of this work is to build a complete and compact Ethernet capable front-end module integrating the detector, the analog front-end chip, and the digital control part including an Ethernet controller. The proposed fully integrated micro-system is shown in figure 3. From left to right the HPD, the Front-end integrated circuit, and the digital part with the Ethernet controller.

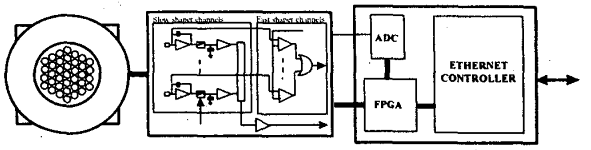

Figure 3: Complete Ethernet Capable Front-end module.

We have designed a first prototype of an Ethernet Capable Front-end Module using an Ethernet device from Agilent Laboratories [2] (Webplug or BFOOT 11501). This device is a custom ASIC based on a 68000 microprocessor and the VxWorks real time kernel. It contains an embedded Ethernet controller, including a Web server and expects to receive the front end data through a 5 Mbps link obeying the IEEE 1451.2 standard [3]. In addition, the BFOOT 11501 provides some very interesting features for physic experiments, such as an UTC time stamping with a synchronization mechanism which allows an accuracy of +/-200 ns through Ethernet and a GPS receiver.

A. *Overview of the IEEE 1451.2 standard*

The IEEE 1451.2 standard provides the ability to produce network-capable smart sensors and actuators. The aim of this standard is to build smart sensors fully independent of any field network. It defines a standard digital interface with hardware lines, protocols and timing, a standard set of functionality for triggering, interrupts, status and control, a calibration model and a representation of physical units. The IEEE 1451.2 standard partitions a smart device into a Network Capable Application Processor (NCAP) and a Smart Transducer Interface Module (STIM). This functional partitioning is shown in figure 3. The NCAP (i.e. the BFOOT 11501) is a processor which interfaces the STIM (i.e. front-end, ADC) with the Network and allows to control the STIM transparently through the network via the Transducer Independent Interface (TII).

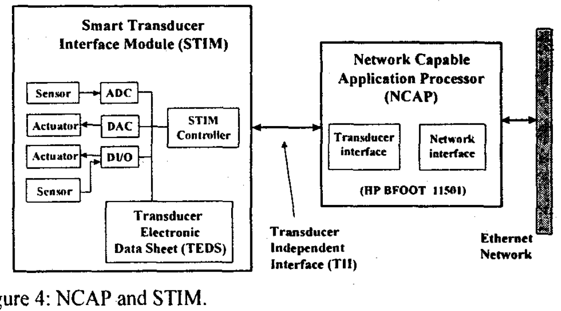
Smart Transducer Interface Module (STIM)
Sensor
ADC
Actuator
DAC
Actuator
DI/O
Sensor
STIM Controller
Transducer Electronic Data Sheet (TEDS)
Transducer Independent Interface (TII)
Network Capable Application Processor (NCAP)
Transducer interface
Network interface
(HP BFOOT 11501)
Ethernet Network

Figure 4: NCAP and STIM.

The TII defines the physical link between the STIM and the NCAP. The TII is built around a synchronous serial communication based on the Serial Peripheral Interface (SPI) protocol and defines a set of triggering functions for reading/writing from/to a transducer. The STIM contains all the functions needed to transform a simple transducer to a smart transducer. It includes a Transducer Electronic Data Sheet (TEDS). The TEDS is a data sheet written in electronic format. It defines the number of channels, the transducer functional type (sensor, actuator, buffered sensor, Data Sequence sensor, Event sequence sensor), calibration model, timing restriction, and any other data needed to fully describe the functionality of the transducer channels implemented in the STIM. The TEDS allows a self identification of the transducers implemented in the STIM. STIM Devices connected to the IEEE 1451.2 interface are made available on the network through an Application Program Interface (API) defined by a set of Universal Resource Locator (URL). In the following we present how the BFOOT was used to design a first prototype of Ethernet Capable Front-end Module.

B. IEEE 1451.2 Network Capable Front-end Module

A block diagram of the first prototype of the Network Capable Front End Module is shown in figure 5. This modules includes 2 high speed ADC (12 bits 25 MHz) for the charge digitization. The time stamping function can be realized in two ways. Either the BFOOT provides a local clock with a synchronization mechanism which allows to synchronize all the nodes through the network with an accuracy of +/- 200 ns, or a dedicated input from an externally distributed clock synchronized on a GPS receiver permits a time stamping with a 5 nanosecond accuracy.

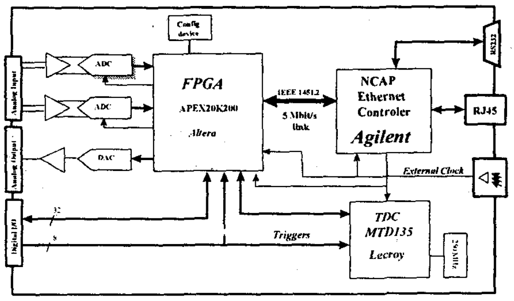

Figure 5: Ethernet Capable Front-end Module.

This first prototype also includes a Lecroy 8 channels mult-hit TDC [4]. The BFOOT is mounted on a daughter board and plugged onto the module. A FPGA Altera APEX 20K200 is used to implement the STIM and interface the NCAP (BFOOT) with the front-end readout sequencer.

C. STIM Architecture

The STIM architecture is shown on figure 6 and includes two main blocs. The first bloc implements the minimal mandatory part of the IEEE 1451.2 standard and contains four sub-modules. The first sub-module includes the TEDS ROM. The second sub-module contains the mandatory status and interrupt mask registers attached to each channel. The third sub-module implements the low level TII interface. The last sub-module is the STIM controller which polls for a BFOOT request and decodes the functional addresses sent by the BFOOT. It also manages the addresses for writing and reading the transducer channels.

The second main bloc contains the transducer channels and the front-end sequencer. This bloc provides all the features needed to control the analog-front chip.

For this first prototype we use two analog front-end chips (VA-TA32CG) from IDE-AS [5] to form a 64 channels front-end device. Its architecture contains 32 slow shaping channels for the charge measurement and 32 fast shaping channels for the trigger generation. When a trigger is detected, an hold signal is applied with a delay corresponding to the peaking time of the slow shaper. The readout of the 64 channels is made through a multiplexer and it is controlled by a shift register. In calibration or test mode two shift registers allow the charge injection at the preamplifiers inputs. A shift register is also dedicated to the DAC codes for the threshold adjustment of each channel, and allows to disable the trigger of any channel. These functions are quite generic and concern any typical front-end device.

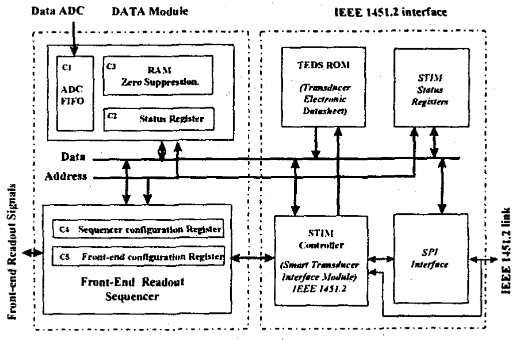

Figure 6: FPGA implementation of STIM architecture..

In order to implement all the front-end functionalities and to make them available through the network, five IEEE 1451.2 channels have been defined.

The first channel is dedicated to the events (ADC, Time Stamp). This channel data format is defined as a burst of 512 32-bit long words corresponding to half the FIFO depth . On a reading request the STIM polls the half full flag of the FIFO before asserting an acknowledge signal and sending the 512 events to the BFOOT.

The second channel contains a general purpose status register for debugging.

The third channel is a 64 x 16 bits RAM which contains the zero suppression thresholds for each channel. These constants are computed after each pedestal run.

The fourth channel is the configuration register and allows to setup the operating mode of the front-end sequencer.

Four modes have been implemented. In the auto-trigger mode, the sequencer polls the front-end trigger signal, enables the ADC, and fills the FIFO according to the zero suppression status flag. The 3 last modes are dedicated to calibration (pedestals, oscilloscope, or peak mode).

The fifth channel is dedicated to the front-end configuration. This is a register which includes the DAC codes for the threshold adjustment of each channel and the bit selection for disabling some channels.

D. Read access method and timing

A node can be accessed in a variety of ways. The primary method is through the web interface by typing a Universal Resource Locator (URL). The figure 7 shows an example of JAVA applet which can read continuously the ADC channel in an oscilloscope mode.

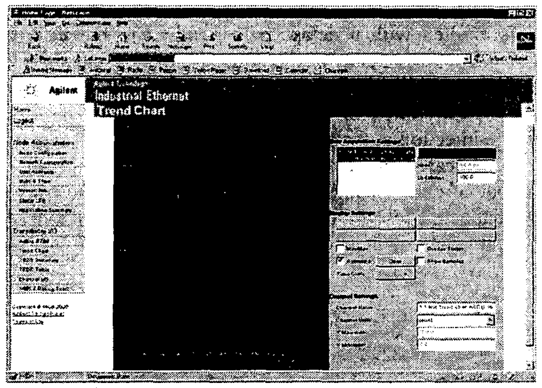

Figure 7: ADC data access via a web browser.

The node can also be accessed by http commands. Here is an example of command for reading a channel defines with 2 bytes of data. The host name of the node is lyotmp9.

Http ://lyotmp9/bin/1451dot2/read ?startChan=2&stopChan=2&stim=1

The corresponding timing at the TII (Transducer Independent Interface) level is shown in figure 8. The NTRIG signal is send by the NCAP to request a sensor reading or writing to an actuator. The NACK signal is send by the STIM to the NCAP to acknowledge a trigger request or a byte transfer. The NIOE signal is send by the NCAP to the STIM to start a transaction frame. The DCLK signal is a clock send by the NCAP to synchronize the bit transfer. First when the BFOOT receives the above http command, it asserts the NTRIG signal (Active low). The STIM which is continuously polling NTRIG and NIOE, asserts NACK if the data are available. The BFOOT also use this signal to time stamp the data with the UTC time and a resolution of 25 ns.

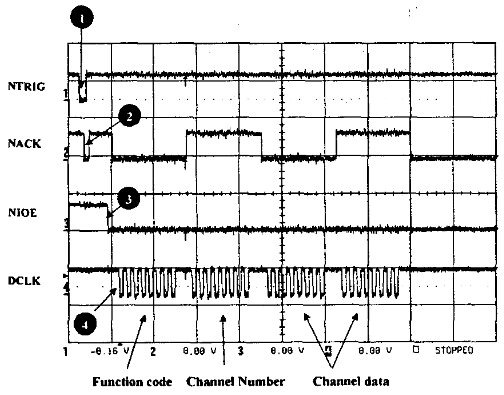

Figure 8: Read timing for a two bytes channel.

In step 3, the BFOOT then asserts NIOE, start clocking DCLK (step 4) and places the data on DIN (Data input of the STIM), the first byte which is transfer is the function code and the second byte is the channel number for which the function must be apply. The next 2 bytes are the data of the channel send by the STIM. The NACK signal is toggled to acknowledge each byte and finally the NIOE signal is released at the end of transaction. The TEDS allows to specify the timing parameters for example the DCLK frequency which is 3 MHz in our case. The BFOOT also provides a read burst command to manage the channels with data repetition. This is the case of our event channel. This channel is defined as 512 events of 32 bits so each burst will contain 2048 bytes. The read burst timing is shown in figure 9.

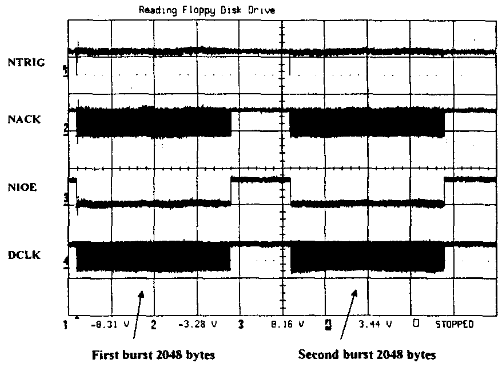

Figure 9: Read burst timing, 2 bursts of 2048 bytes.

The read burst function implements post-processing on each datum before sending it on Ethernet. This post processing includes data conversion and a correction engine, which is time consuming. In order to optimize the speed for the OPERA experiment, we have defined a data streaming application which allows to keep a network connection opened and send the data continuously on Ethernet.

E. Data streaming application

This data streaming mode is a specific application (figure 10) based on two threads, which share the same set of ring buffers. The writer thread setup an IEEE 1451.2 transaction at a period specified in parameter and fills one of the buffers.

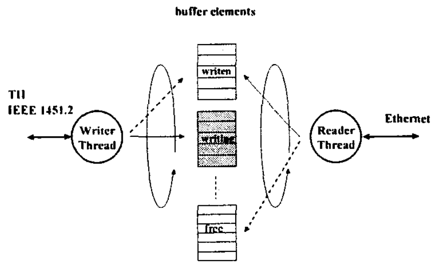

Figure 10: Data streaming application.

When the ISR (Interrupt Service Request) is generated the write Thread switches to the next buffer. The reader thread ships the buffer down to the Ethernet socket.

The writer thread should be run at high priority. In case of TCP/IP congestion the reader thread can slow down and data can be lost if the writer thread fills the entire buffer. Having a large number of buffer elements in principle reduces or eliminates any gaps in the data.

The PC application will need a similar multi-threaded architecture. A reader thread will be attached to each node. Each reader thread will grab an empty buffer from the free buffer list, will read from the socket and will place the buffer on full list. The database thread will grab a buffer from the full list, will write it to disk and will place it back to the empty list.

We have made a first test with a sampling period of 20 ms and a burst size of 2048 byte, each burst containing 512 events of 32 bits. The BFOOT sends the data continuously to the client host. The data rate is about 800 Kbits/s which corresponds to 400 events per second and per channel at the front-end inputs. This is largely sufficient for our data rate, the final data rate per channel is expected to be at least an order of magnitude less.

IV. CONCLUSIONS

The standardization of Ethernet, its industrial availability and the continuous upgrade of its performances, make the use of Ethernet as unique network for data transmission and slow control conceivable. While the concept of the Ethernet Capable Front-end Module is widely used in the case of slow-control type sensors (e.g. pressure and temperature gauges), we have extended the concept to include the data-acquisition requirements of a high-energy physics experiment.

We have designed and built a first prototype in order to prove the feasibility and evaluate the real performances of such a system. It communicates with the sensor through the IEEE 1451.2 standard. The STIM has been implemented in VHDL and synthesized in a FPGA. It includes all the necessary functions for a multi-channel photo-multiplier sensor interface. Embedded slow control, monitoring and readout functions are directly accessible from the network by TCP/IP or a simple WEB browser.

It allows to validate the front-end readout through Ethernet, using an embedded Ethernet controller (BFOOT 11501). Data rates close to 1 Mbps have been achieved.

Unfortunately, Agilent has decided to discontinue this chip due to insufficient demand. We luckily have a sufficient quantity to test the above ideas in a network composed of at least 16 nodes, that is a typical X-Y plane .

Furthermore, this STIM can be used with any other NCAP which implements the IEEE 1451.2 standard. Therefore, another interesting perspective, possibly extending the data rate bandwidth, is the use of new real-time JAVA processors [6]. The idea of building a full JAVA data acquisition system from the front-end electronics to the event building data base is very attractive [7],[8].

V. REFERENCES

[1] M. Guler et al., CERN/SPSC 2000-028, SPSC/P318, LNGS P25/2000, Jul. 10 2000.

[2] http ://www.agilent.com

[3] Standard for a Smart Transducer interface for sensor and actuators - Transducer to Microprocessors communication protocol and transducer electronic data sheet (TEDS) format. 1997- ISBN 1-5593-7963-4.

[4] LeCroy, Technical Data, TDC MTD135, MTD133B 8 channels multi-hit, time-to-digital converter.

[5] http ://www.ideas.no

[6] Two exemples of such JAVA processors are for instance proposed by AJILE (www.ajile.com) and PATRIOT (www.ptsc.com)

[7] Y.Yasu et al., Prototype Performance of distributed DAQ using HORB based on Java. The 10th IEEE Real time Conference 97, BEAUNE, FRANCE, 22-26 September. (http :// www.online.kek.jp/~online/JavaDAQ/)

[8] Java for Embedded systems. A special programme at the Embedded system show. Olympia, London, May 24th-25th, 2000-04-12.