The DAQ-Column prototype of the CMS experiment

G. Antchev [a], E. Cano [a], S. Cittolin [a], S. Erhan [b], D. Gigi [a], P. Gras [a], J. Gutleber [a], C. Jacobs [a], F. Meijers [a], E. Meschi [a], A. Ninane [c], L. Orsini [a], L. Pollet [a], A. Racz [a], D. Samyn [a], P. Sphicas [a,d], P. Scharff-Hansen [a], C. Schwick [a,1]

[a] *CERN, 1211 Geneva, Switzerland*
[b] *University of California Los Angeles, USA*
[c] *Université catholique de Louvain-la-Neuve, Louvain-la-Neuve, Belgium*
[d] *Massachusetts Institute of Technology, USA*

***Abstract*—The data acquisition system (DAQ) of the CMS experiment must collect, for each event accepted by the level-1 trigger system, the data fragments produced by approximately 500 different data sources. After assembling these fragments into whole events, a processor farm will be used to analyze the data in order to determine its relevance for physics analysis. Finally, the interesting events will be written to permanent storage media where they will be available for physics analysis. Given the average size of 2 kB for the fragments, and a maximal trigger rate of 100 kHz, the mean bandwidth of the system must sustain 100 GB/s. Such performance can be achieved by a parallel system architecture utilizing a fast network switch.**

The system currently being developed consists of both custom designed hardware modules based on FPGAs and commercial components like PCs or network switches. Various functional blocks can be implemented in hardware or software. A vertical slice of the system, the DAQ-column, is being built in order to study different implementation possibilities at all stages of the system. It will form the backbone for the development of the final system, which will be produced in the year 2004.

I. Introduction

The requirements for the Data Acquisition System (DAQ) of the CMS experiment follow from the operating conditions of the LHC accelerator on one side and from the physics goals of the experiment on the other side [1].

The LHC will achieve a Luminosity of 10^{34} $1/(Hz*cm^{-2})$ with a bunch crossing frequency of 40 MHz. The rate of proton collisions in the CMS detector will be around 1 GHz, implying that several proton interactions occur at every bunch crossing. The first level trigger uses dedicated fast available data of the calorimeters and the muon detectors in order to reduce the rate of incoming data from 40 MHz to a maximum of 100 kHz by selecting events with signatures indicating interesting physics. The average event size is expected to be 1 MB. Therefore the DAQ system has to sustain an average data flow of 100 GB/s.

To achieve this sustained bandwidth data are processed in parallel at all stages of the DAQ system. In the following section we present the general architecture of the DAQ followed by a definition of the DAQ column. Subsequent sections present in more detail the functional components of the DAQ column, possible implementation choices and finally a summary of the current status of the development.

II. The CMS DAQ-System

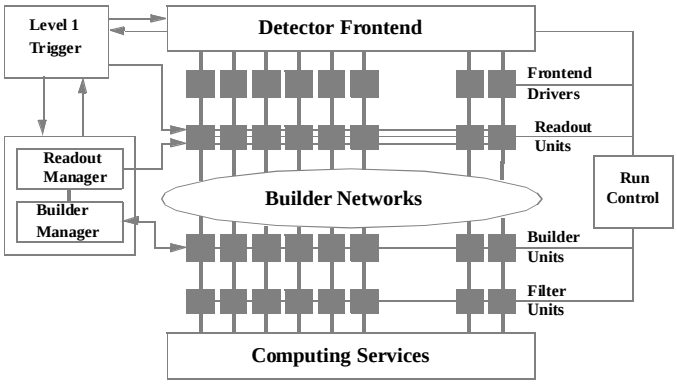

Fig.1. A blockdiagram of the CMS DAQ system. In the center the components involved in the data transfer are shown. The blocks on the left of the figure (Level 1 Trigger, Readout Manager and Filter Manager) control the event building process. The Run Control system allows the operator to interact with the system.

A block diagram of the CMS DAQ System is shown in Fig. 1. The central part shows the flow of the data. The various subdetectors store data for each bunch crossing in pipeline buffers. On each first level trigger the data of the corresponding bunch crossing are read out and the resulting event fragments are fed into ≈500 readout systems.

[1] Corresponding author; email address: Christoph.Schwick@CERN.CH

Fragments corresponding to the same event are sent via the Builder Networks to the same filter system (this is the “event building” task). There the decision of the High-Level trigger is taken reducing the event rate in two steps to 100 Hz. The finally selected events are sent to the computing services.

The system is driven by the first level trigger, which initiates the detector front-end to push their event fragments into the readout systems (Front End Drivers: FEDs). In addition the trigger decision is sent to the Event Manager composed of a Readout Manager and a Builder Manager. These two entities control the event building process as will be explained in the following section.

III. The DAQ Column

The DAQ Column is defined as all the components, which a data fragment traverses after leaving the detector front-end and before entering into the computing services. It can be seen as a vertical slice through the DAQ system. Fig. 2 shows how these components are arranged. The following sections describe the functionality of each of these components and discuss various implementation possibilities. [2]

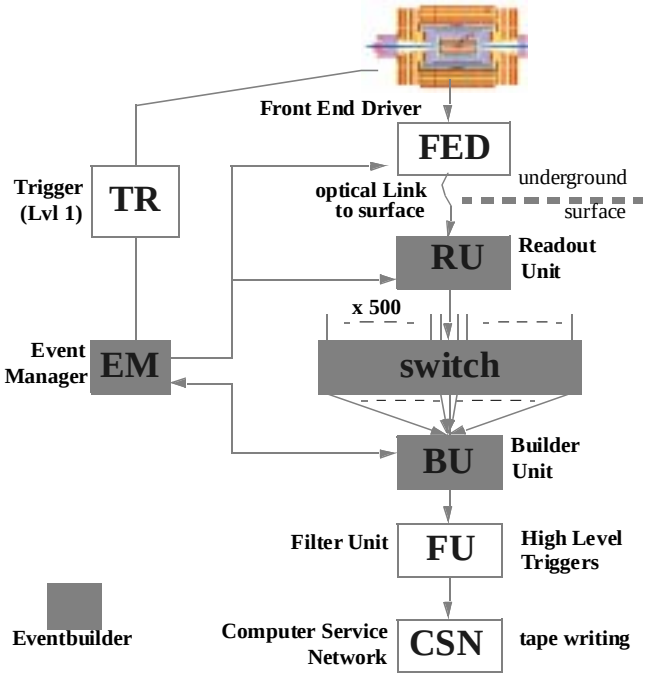

Fig.2. Blockdiagram of the DAQ-Column. Shaded components are involved in the event building process.

A. The Front End Driver (FED)

The Front End Driver is the last detector-dependent component in the data flow. Here event fragments are formatted by adding to the event data a header and a trailer containing event relevant information needed during event building and consistency checks. The way the FED writes its data into the DAQ system is defined in the SLINK64 protocol [4], [5]. From the FED point of view, the SLINK protocol is equivalent to writing into a FIFO. The hardware of the SLINK will be plugged onto the FED as a mezzanine card. From there data are transferred to a Merger card.
Since not all FEDs generate 2 kB data per event in the Merger, up to three SLINK sources can be merged. The Merger itself is a PCI card in a PC controlled by the DAQ system. This setup allows testing the entire DAQ system without relying on the availability of the FEDs.

B. The DAQ-Link

The event fragments of the FEDs have to be transported over an approximately 100 m long link to the surface building where the Event Builder is housed. This link has to sustain 200 MB/s on average. In order to absorb bursts of high data rates its throughput is specified at 400 MB/s. This avoids the need of large buffers in the FEDs.

A prototype of a DAQ Link has been built and is currently being tested. It uses a Vitesse VSC2714 chip and four Infineon V23818-K305-V15 electrical to optical converters. In total 8 fibers form a full duplex link with 400 MB/s for each direction. The prototype of this link is shown in Fig. 3.

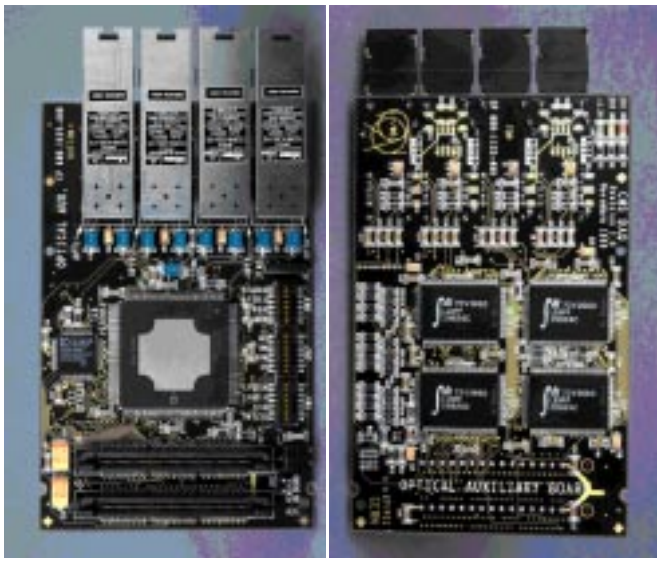

Fig. 3. The prototype of the DAQ-Link. Component side and solder side of the card are shown. On the top of the figure the four optical converters can be identified.

As an alternative to this custom design a fully commercial solution is currently being considered. A possible candidate is the optical link offered by Myricom. It reaches a bandwidth of 2 Gb/s for the payload incorporating a simple protocol with flow control so that no packet loss can occur.

[2] The Builder Network itself is of course also a component traversed by the fragments. Since a vertical slice through the DAQ system cannot test the performance of this component, it is not considered part of the DAQ-column. More information on the tests with different switch technologies can be found in [2], [3].

Although not directly a component of the DAQ-column, also the Event Manager is described here since its functionality is required in order to operate the DAQ-column.

C. The Event Builder (EVB)

The shaded components in Fig. 2 are involved in the event building. The process is controlled by the Event Manager (EM). All events in the Event Builder are identified with a unique label, the "event-ID" or ID for short. The primary responsibility of the Event Manager is the assignment of IDs to newly triggered events, as well as the assignment of an event in the Readout System to an available Builder Unit. The EM is therefore split into two logical blocks (the Readout Manager, RM, and the Builder Manager, BM) corresponding to the above functions.

On each Lvl1 trigger the RM takes an ID from the pool of available IDs and broadcasts it to all Readout Units (RUs). The next incoming event is associated with the new ID and buffered in the Readout Unit Memory (RUM). On the other side of the builder switch the Builder Units (BUs) have to assemble entire events. As soon as they have free resources to process a new event they send a request to the BM. The BM acknowledges by sending back an ID, which has previously been broadcasted, to the RUs. The BU then requests event fragments, which correspond to the received ID, from all RUs. After the event has been assembled, it is sent to a Filter Unit (FU). The BU then releases the ID by sending a message to the BM, so that it can be put back into the pool of available IDs.

Various implementation options are considered for the components of the Event Builder.

1) The Readout Unit

Fig. 4 shows the Readout Unit which consists of three blocks: the Readout Unit Input (RUI), the Readout Unit Memory (RUM), and the Readout Unit Output (RUO)[6]. The RUI receives the event fragments from the DAQ-Link and buffers them until the RUM requests them. The buffer must be sufficiently large in order to absorb the latency measured from the arrival of the first level trigger at the EM to the arrival of the event ID at the RUM. The RUM reads fragments from the RUI in the same sequence as they arrive and saves them, together with the ID, in its internal memory. The requests of the BU for events enter the RU via the RUO. The fragment is then sent over the switch to the requesting BU by the RUO.

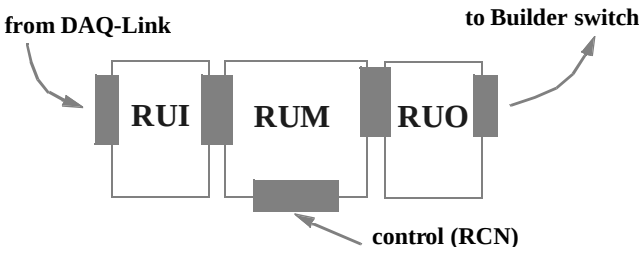

Fig. 4. The functional blocks of the Readout Unit.

The current implementation of the RUI is a PCI mezzanine card, which carries a connector to plug in the DAQ-Link. An FPGA controls the link and moves the incoming data into a 32 MB buffer. From there the fragments are transferred to the RUM as soon as a request message from the RM arrives.

The RUM is currently implemented as a PCI card with three PCI busses [6]. One bus connects to the RUI, the second to the RUO. The third one is used to set up and control the RU. In the current setup the requests of the RM also arrive via this bus. A commercial Network Interface Card (NIC) implements the RUO. The setup is currently being tested.

Alternatively the RU functionality could be implemented in software using a PC housing two separate PCI busses with 64 bits / 66 MHz in order to cope with the high bandwidths of the input and the output. A schematic diagram of such a setup is shown in Fig. 5. PCs of this type are not yet available on the market. Currently tests are performed on configurations containing an independent 64 bit / 66 MHz and a 32 bit / 33 MHz PCI bus.

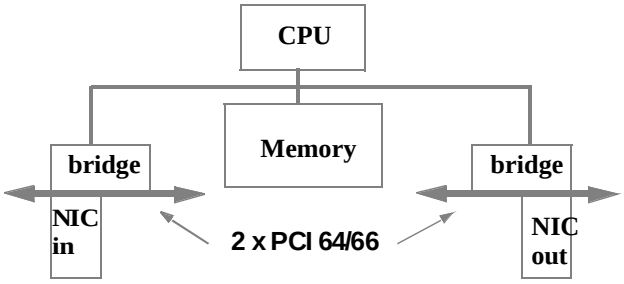

Fig.5. Possible PC setup for the implementation of the RU functionality.

2) The Builder Unit

The Builder Unit needs to assemble the event fragments from the RUs via the builder network. The demands on the bandwidth are equivalent to those for the RU. A prototype based on a Power PC has been built. It contains a 64 bit / 66 MHz PCI bus and implements a high-speed link according to the RamLink specifications [7]. In order to form a BU two modules are connected via the RamLink. The first module is equipped with a NIC card connecting to the builder switch and serves as the input. The second card interfaces to the Filter Unit (FU) via another Network Interface.

3) The Event Manager

The functionality of the Event Manager has been explained above. Currently an implementation using a single PC is being evaluated. In order to broadcast IDs to the RUs, both Gigabit Ethernet and Fire Wire (IEEE 1394) are being investigated. In addition, a link to the First Level trigger has to be designed, since the Event Builder needs some additional information for each triggered event. Such information could be used to add some additional data routing options in the EVB, eg. to transfer a specific event type (eg. calibration events) to a dedicated group of BUs.

IV. SOFTWARE

The CMS DAQ-system is a large distributed system consisting of thousands of functional components operating in parallel. It is heterogeneous since the different components are either implemented in custom hardware or in software on standard computers or a combination of both. In order to create a homogeneous environment for the software involved, a framework for distributed DAQ-systems (XDAQ) has been developed [8]. It creates an environment for DAQ specific applications involved in the transfer or processing of event data. It also allows configuring and controlling the DAQ system. Since the framework is independent of a specific DAQ-architecture it can be used in large distributed systems as well as in small systems like the DAQ-column prototype.

At startup the XDAQ executables on each node process an XML description of the specific DAQ system in order to start and configure the applications involved [9]. XDAQ controls the system by sending SOAP [10] messages to selected components, which invoke specific actions on the target (remote message invocation).

The framework implements an event driven middleware architecture [11] inspired by the I2O specification [12]. This means internal application events (eg. timeouts, DMA interrupts) as well as commands from other XDAQ applications received over the network, are homogeneously formulated as I2O message frames.

Different transport protocols for data transfers are supported (raw Ethernet, TCP/IP, HTTP/SOAP and Myrinet). The framework is currently running on Linux, Solaris, VxWorks and MaxOSX.

V. SUMMARY

The CMS data acquisition system needs to process the event data at the rate of the first level trigger (max. 100 kHz). Approximately 500 data sources generate fragments of 2 kB which are sent via optical links to the Event Builder which assembles the events before sending them to the higher level triggers. This task is performed by a system in which many functional components work in parallel in order to cope with the high data rate of 100 GB/s. A vertical slice of this system called the DAQ-Column is currently being built. Its functional components have been described and the various implementation options discussed. Within the next year DAQ-Column will be used as a test bench to test the various implementations in order to finalize the design.

VI. REFERENCES

[1] The CMS collaboration, "The Compact Muon Solenoid," CERN, Technical Proposal No.7, *LHCC 94-38*, Dec. 1997.

[2] G. Antchev et al.., "The CMS Event Builder Demonstrator based on Myrinet", 14th *IEEE NPSS Real Time Conference*, Santa Fe, New Mexico, USA, June 1999.

[3] M. Bellato et al., "The CMS Event Builder Demonstrator based on GigaEthernet Switched Network," *CHEP2000*, Padova, Italy, Feb. 1999.

[4] O. Boyle, R. McLaren, E. van der Bij, "The S-LINK Interface Specification". CERN, Switzerland. [Online]. Available at http://hsi.web.ch/HSI/s-link/spec/

[5] A. Racz, R. McLaren, E. van der Bik, "The S-LINK 64 bit extension specification: S-LINK64". CERN, Switzerland. [Online]. Available at http://hsi.web.ch/HSI/s-link/spec/

[6] G. Antchev et al., "Readout Unit Prototypes for the CMS DAQ System," *LEB2000*, Cracow, Poland, Sep. 2000.

[7] IEEE Std 1596.4-1996, "IEEE Standard for High-Bandwidth Memory Interface Based on Scalable Coherent Interface (SCI) Signaling Technology (RamLink)"

[8] J. Gutleber, E. Cano, S. Cittolin, F. Meijers, L. Orsini, D. Samyn, Architectural Software Support for Processing Clusters," *IEEE International Conference on Cluster Computing*, Nov. 2000.

[9] See for example W3C site "Extensible Markup Language (XML)," W3C. [Online]. Available at http://www.w3.org/XML/

[10] D. Box et al., "Simple Object Access Protocol (SOAP) 1.1," W3C, 8 May 2000. [Online]. Available at http://www.w3.org/TR/SOAP/

[11] M. E. Fiucynski and B, N. Bershad. "An extensible protocol architecture for application-specific networking." In *Proceedings of the USENIX 1996 Annual Technical Conference*, January 1996.

[12] The I2O Special Interest Group, "Intelligent I/O (I2O) Architecture Specification," Version 2.0, march 1999. [Online]. Available at http://www.intelligent-io.com/