Validation of the ATLAS Trigger/DAQ Network architecture using hardware data emulators

M. J. LeVine, S. Stancu, Ch. Haeberli, L. Tremblet, R. Beuran, C. Meirosu, R.W. Dobinson, B. Martin, E. Knezo, H.-P. Beck, R. Hauser, D. Botterill

***Abstract*-- Hardware data emulators are used to deploy a large-scale model of the ATLAS Data Acquisition architecture. The emulators, based on FPGAs and on the Alteon Gigabit Ethernet NIC, are described, and their performance determined. The emulators are used in the large-scale test bed; sample results are presented.**

I. Introduction

The ATLAS Trigger/DAQ (T/DAQ) architecture involves ~1600 data sources (Readout Buffers, or ROBs) connected to several hundred Level 2 processing units and at least 100 SubFarm Interfaces (event builders). An Ethernet network connects all of these nodes. As shown in Fig. 1, there are 3 layers of switches necessary to accommodate the number of ports required [1]. A number of questions arise which affect both software implementation and hardware deployment, including: frequency of packet loss, side effects of flow control, effectiveness of traffic shaping, overheads of different protocols, utility of VLANs in limiting scope of broadcasts and in preventing loops which would be discarded by the spanning tree protocol.[2]. The answers to some of these questions can only be obtained unambiguously by building the final system and running it, but the cost would be prohibitive. Instead, we have chosen to build a test bed, which is a slice of the final system comprising about 10 per cent of the final nodes, on which to obtain answers to these questions. Deploying such a system requires ~150 ROBs; the ROBs are not presently available and will not be available in these numbers for at least one year. Thus we have chosen to use hardware data emulators which behave as either ROBs or ROSes (defined in the next section) so that the data flow in the test bed system is as much like that in the final system as possible.

We describe the emulators and the efforts made to understand the degree to which the emulators themselves contribute to limited event rates and lost packets. Finally we describe first results in the test bed using the emulators.

Manuscript received June 16, 2003. This work was supported in part by the U.S. Department of Energy under contract no. DE-AC02-98CH10886.

M.J. LeVine is with Brookhaven National Laboratory, Upton, LI, NY 11973 (e-mail: levine@bnl.gov).

S. Stancu, R. Beuran, and C. Meirosu are with CERN, 1211 Geneva, Switzerland, and "Politehnica" University of Bucharest, Bucharest, Rumania

Reiner Hauser is with the Michigan State University Department of Physics and Astronomy, East Lansing, Michigan, US

Christian Haeberli and Hanspeter Beck are with the Laboratory for High Energy Physics, University of Bern, 3012 Bern, Switzerland.

R.W. Dobinson , B. Martin, L. Tremblet, and E. Knezp are with CERN, 1211 Geneva, Switzerland.

D. Botterill is with Rutherford Appleton Laboratory, Chilton, Didcot, Oxon, UK

II. ATLAS TDAQ architecture

The ATLAS TDAQ domain begins with a set of approximately 1600 data sources, called Readout Buffers (ROBs), which receive event fragments from the detector Readout Drivers (RODs) via fiber Readout Links (ROLs), on each Level 1 (LVL1) accept; LVL1 accepts occur at rates of up to 75 kHz. The event fragment transmitted over each ROL are nominally up to 1.4 kB.

An event is assigned to a member of the Level2 processing farm (L2PU), which requests event fragments associated with this event from selected ROBs, depending on the nature of the LVL1 information for this event. On average the number of ROBs addressed will be <2% of the total.

The L2PU applies selection criteria to the event based on the event fragments received. A decision is passed to the Dataflow Manager (DFM). If the event is accepted by LVL2, the DFM assigns it to a SubFarm Interface (SFI), whose task is to build the complete event and pass it to a member of the Event Filter (EF), the next step in the process of deciding whether to write the event to tape or other media or to discard it. The SFI assigned to this task requests fragments from all of the ROBs, and uses the responses to build the event before passing it to the EF.

There are several versions of the TDAQ organization under active consideration: one version considers all of the ROBs to be independently connected to the TDAQ network, and the nodes L2PU and SFI communicate directly with each ROB. Other versions have the ROBs grouped into a Readout System (ROS), still with independent network interfaces, and which communicate with a PC called the ROS controller via either Ethernet or PCI bus. In these versions, the L2PU and the SFI communicate with the ROS controller, and the responses are aggregated into a ROS response issued by the ROS controller.

III. ROS emulators

Two ROS emulators have been implemented: an FPGA Fast Ethernet ROB emulator and an Alteon Gigabit Ethernet ROS emulator. Both are described in detail in the following paragraphs. For the studies reported here, the ATLAS Data Collection Message Passing [3] was implemented as a

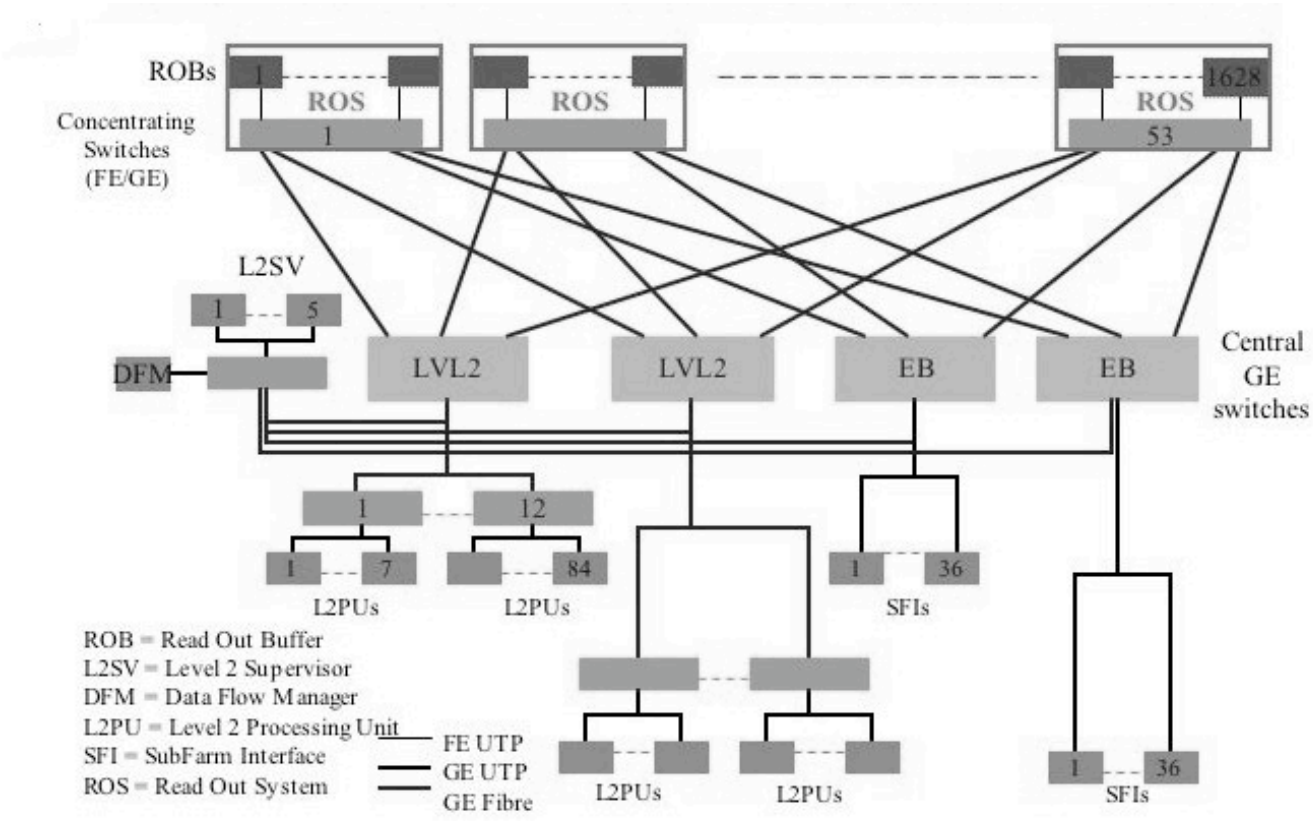

Fig. 1. The ATLAS Dataflow architecture

lightweight layer on raw Ethernet, whose functionality is similar to that of UDP.

Both types of emulators can operate in alias mode, where a given emulator responds to multiple node IDs using the same Ethernet hardware address. This allows the simulation of a large number of nodes using a smaller number of physical emulators, up to the limit of the emulators' ability to keep up with the request rate.

A. FPGA emulator

The FPGA ROS emulator is based on a Fast Ethernet (FE) tester constructed at CERN as a general-purpose traffic generator and device for measurement of network parameters (e.g., latency, packet loss, throughput). The FE tester implements 32 FE ports. It has been described in detail [4,5].

The ROS emulator uses the physical platform of the FPGA emulator, with the FPGAs programmed to respond to data requests coming from ATLAS Data Collection. The newly coded FPGAs communicate using a message format conforming to the ATLAS Data Collection Message Passing specifications, and produce data formatted according to the ATLAS Event Format [6]. The contents of the detector data payload produced by the emulator are meaningless; only the length of the response is intended to be meaningful for the purposes of these measurements.

Incoming requests are parsed by one process on the FPGA and translated into a 32 byte descriptor, which is written to a queue serviced by the concurrent process responsible for building the outgoing response frames. The ROS emulator issues flow control packets to protect its descriptor queue, and responds to flow control packets received. It also is capable of generating VLAN tags.

Squeezing the ROS emulator functionality into the modest-sized FPGAs chosen for the FE tester, Altera FLEX 10K50 (50,000 gates), proved to be a challenge. A ROS emulator functionality was achieved by use of on-chip memory to build a template of the outgoing packet headers and trailers. About 90% of the words in the various headers are invariant for all frames; only the remaining 10% have to be written to the memory on a case-by-case basis. The contents of the memory were then used to stream to the Ethernet PHY.

In principle, the data responses from a ROS can comprise multiple RODs, but the restrictioins imposed by the limited resources of the FPGA do not allow this; only single ROD responses can be generated. Additionally, the limited resources available prevent the FPGA emulator from implementing the UDP protocol; only raw Ethernet has been implemented.

In spite of the limitations imposed by the FPGA resources, the response of the emulator is extremely fast. The response frame is normally transmitted starting less than 1 μs following reception of the entire request frame. The only conditions under which the delay would be longer are those times when the transmitter has received a flow control PAUSE frame due to congestion elsewhere in the system. The FPGA emulator is capable of receiving and transmitting at Fast Ethernet line speed.

B. Alteon emulator

The GE ROS emulator is implemented on the Alteon programmable NIC, based on the Tigon 2 chip. The modules dealing with the host communication have been suppressed, as they are used only for starting the emulator and gathering statistics. The chip has two embedded MIPS processors (RISC processors) running at approximately 86 MHz, which can randomly access the 1MB memory of the NIC. This memory contains both the binary code and the data structures needed for operation.

The Tigon PCI/Gigabit Ethernet Controller provides a 64-bit high-speed memory bus to connect to local memory. Six entities within the Tigon arbitrate for the use of the Memory Bus. A priority scheme is enforced between these requesters so that no requester can cause a loss or corruption of the data flow. The Gigabit Ethernet interfaces (receive, transmit) have the highest priority within the Tigon, since it is imperative that the Gigabit Ethernet interfaces never under-run or overrun.

Host accesses to the Tigon must occur in a timely fashion and therefore it is next in the priority scheme. A host access takes place only when there is no DMA activity on the bus. The DMA is used only for downloading the firmware to the NIC's memory, and for reading statistics. Therefore, under normal operation of the ROS emulator the DMA engines are not active. The NIC is controlled using a modified Linux device driver, which makes the NIC's memory appear as a regular device for the user level code.

1) Internal processors

The Tigon contains two internal 32-bit general-purpose processors, modeled after the R4000 RISC processor; however the supported instructions and their decoding is unique to the Tigon. Many instructions were removed and several new ones added for embedded optimization.

The operation of the processor is governed by firmware. External to the Tigon is a small non-volatile memory which contains all the power on diagnostics and PCI initialization tasks. Any additional software can be downloaded by the host driver into the local SRAM memory. Since each local memory operation fetches 8 bytes, up to two processor instructions can be loaded into the Tigon on each instruction fetch cycle. A small instruction cache as well as SRAM internal to the Tigon (Scratch Pad) is also provided to enable the processor to reduce its usage of the valuable local memory bandwidth. Firmware is compiled from C and

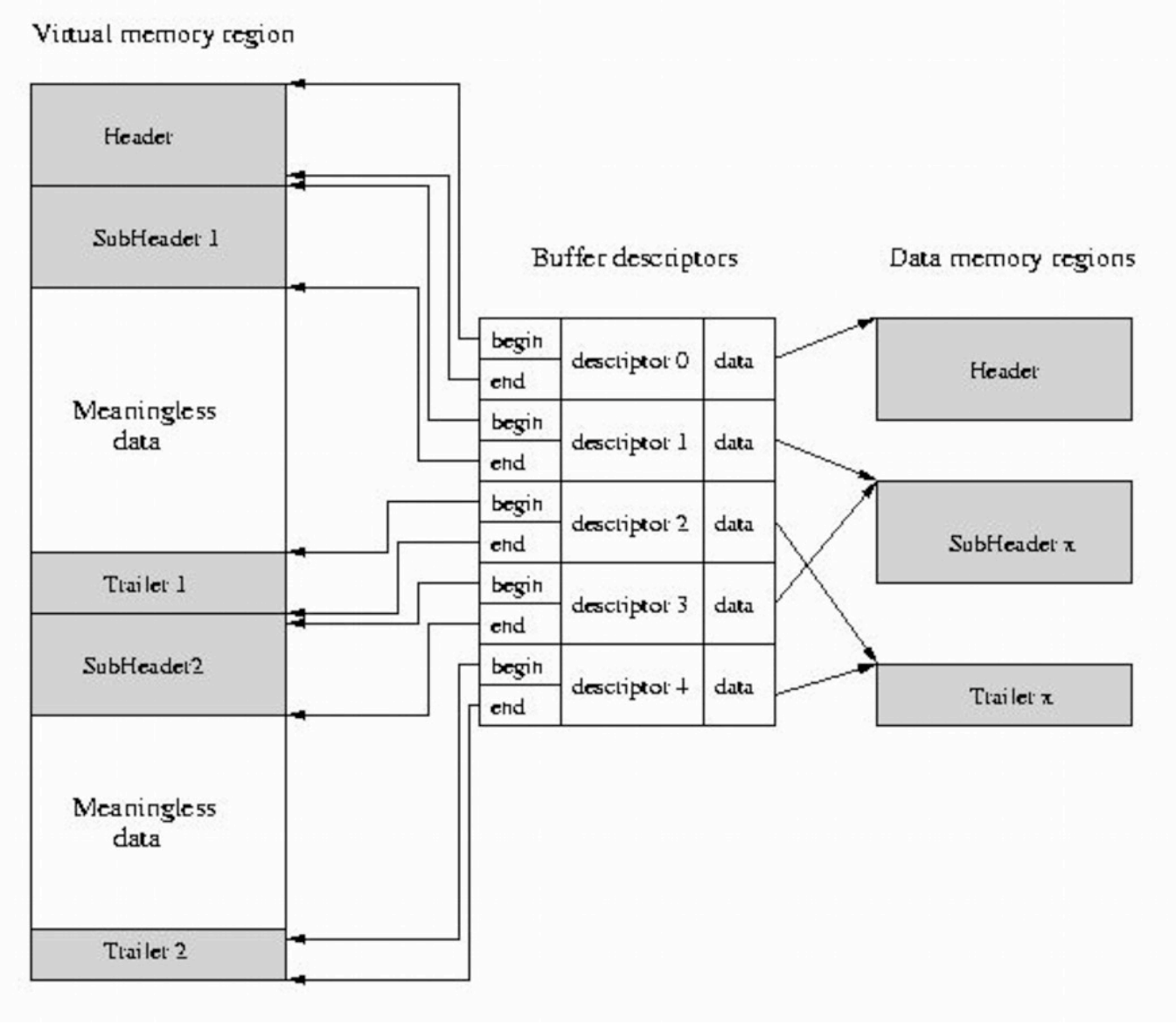

Fig. 2. Virtual buffers used to increase performance in the Alteon emulator

assembly language using GNU R4000 tools.

One processor receives frames, assembles them into messages (a message can be sent using several frames [3]), and extracts the information into a "message descriptor". The message descriptor contains all the information needed for building a reply message. The message descriptors are then written to a shared queue.

The second processor dequeues message descriptors from the shared queue, and builds up the response message. The response message is subsequently split into frames which are passed to the Ethernet transmit interface.

The Tigon 2 chip has a "semaphore" register which allows having shared memory regions, like the descriptor queue, preventing simultaneous access from both processors.

2) Building the ROS fragment

The Alteon emulator is able to generate multi-frame messages up to 64 kB in length. The ROS fragment (the reply generated by the emulator) can contain up to 50 ROD fragments, so long as the maximum size limitation is not exceeded.

The response rate of the Alteon ROS emulator is limited by the CPU frequency and the use of the shared memory Bus. The use of multi-frame messages implies building the message to be sent in a memory region, and then copying pieces from that message in separate Ethernet frames. The process of disassembling the message into Ethernet frames cannot be avoided, but it can be improved. Not all the contents from the message sent by the ROS are relevant (only ROS, ROB and ROD headers are meaningful).

In order to improve the speed of generating the replies, we have used a "virtual buffer" (see Fig. 2). Not all the data in the reply message is relevant, and also some pieces of the relevant data can be identical. The "virtual buffer" is seen from the outside as a contiguous region of memory, representing a mixture of relevant and irrelevant data. The internal representation of the "virtual" buffer consists of a vector of descriptors and the relevant data memory region.

The example in Fig. 2 shows a virtual buffer made up of 5 buffer descriptors and three data memory regions (Header, SubHeader x, Trailer x). Using a proper interface, this virtual buffer appears as the "virtual memory region" on the left side of the figure, when the user wants to copy the contents of this buffer to a different memory location.

The virtual buffer can significantly improve the performance when the amount of meaningless data is large. For example, for a 41-frame message with a single ROB fragment inside:

- building up the memory region and then splitting it into frames requires 192452 clock ticks.
- building up a virtual buffer, and then splitting it into frames requires 36368 clock ticks, for an improvement of factor 5 in speed.

IV. Validation of Emulators

A number of measurements have been made using the emulators in small test setups with one PC issuing requests, one or two switches and a number of emulators. There are from 1 to 8 Alteons and 1 to 127 FPGA emulators present in the various tests. In most of these tests the emulators are operating in alias mode described above. The PC is programmed to issue requests at intervals which should produce a given data rate for the responses. The latency is measured as well as the frequency of packet loss. For these measurements latency is defined as the time interval between delivering the request to the PC operating system (OS) and

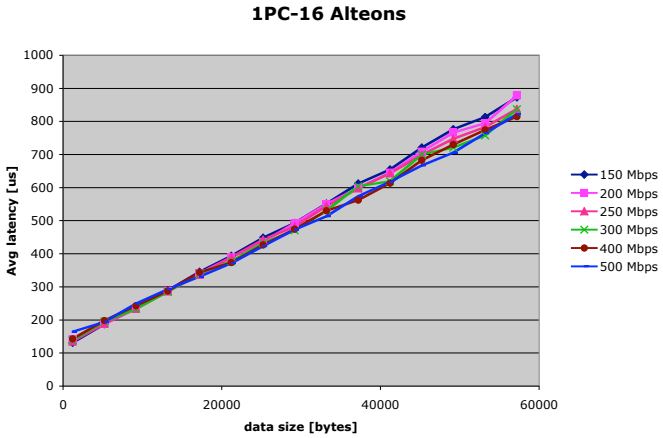

Fig. 3. Latency *vs* data size for several requested data rates, using 16 Alteon emulators

receiving the response from the PC operating system. Thus latency includes OS effects, waiting time in intervening switches, as well as the latency due to the emulator itself.

Fig. 3 shows a latency measurement made with 16 Alteon emulators, where the total bit rate varies from 150 Mbps to 500 Mbps, for data payloads ranging from 1200 Bytes to 60000 Bytes. The latency varies linearly with the payload size, which reflects transit time on the wire and in the switch.

Fig. 4 shows an identical measurement made using a single Alteon emulator. The linear dependence of latency on data size is still the case, though barely visible in this figure, for data rates below 400 Mbps. But for data rates higher than this, the latency increases dramatically, reflecting queues

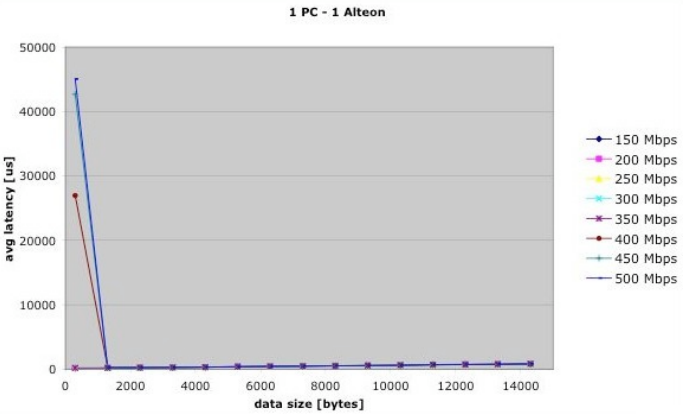

Fig. 4. Latency *vs* requested throughput using 1200 Byte data size, for various configurations of Alteon and FPGA emulators. Numbers of Alteons greater than 16 refer to virtual emulators (see text).

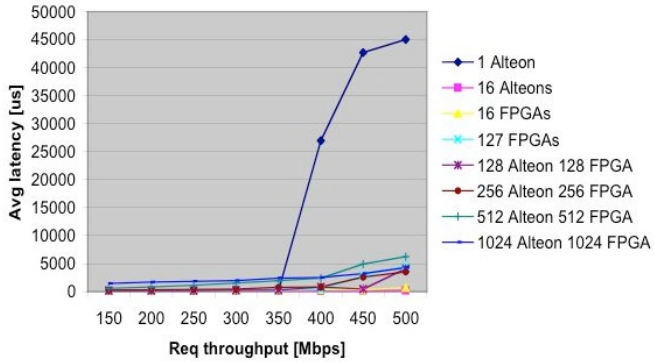

Fig. 5. Latency *vs* requested throughput using 1200 Byte data size, for various configurations of Alteon and FPGA emulators. Numbers of Alteons greater than 16 refer to virtual emulators (see text).

filling in the Alteon. This represents the upper limit of the Alteon's usefulness as a ROB emulator.

Fig. 5 shows a similar measurement made with a variety of emulators, many of them in alias mode, all with a data size of 1200 Bytes. Configurations labeled with numbers of nodes greater than 16 (Alteons) or 127 (FPGAs) were carried out in alias mode. The latency is well behaved except for the case of the single Alteon.

Summarizing these measurements and others not shown here, the emulators behave as expected with the exception of the Alteons at high data rates (>350 Mbps) for the smallest data payloads (1200 Bytes). All emulators have reasonable latencies at any data rate for data payloads > 1200 Bytes.

V. Other Measurements

A variety of studies have been carried out using the emulators, which have been reported elsewhere [7], to determine the behavior of switches using VLANs, flow control, broadcast, and the effective size of the MAC address

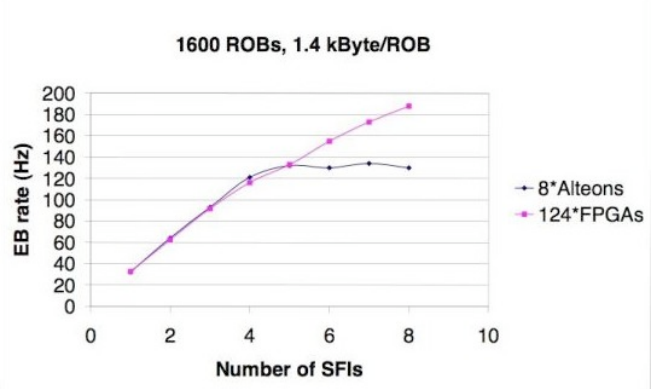

Fig. 6. Event building rate as a function of number of parallel event builders, for 8 Alteon emulators and for 124 FPGA emulators.

table internal to the switches used.

VI. Use of the Emulators in the ATLAS Architecture

The ROB/ROS emulators are now in use in the ATLAS Dataflow test bed. These measurements are ongoing, but some sample measurements are presented here.

Fig. 6 shows results obtained using 8 Alteons or 124 FPGA emulators, responding to requests addressed to 1600 ROBs, building an event of 2.2 MB. The measurement was carried out with the number of event builders (SFIs) working in parallel ranging from 1 to 8. Fig. 6 shows that the gain with increasing the number of event builders is linear with the FPGA emulators, while the Alteon results show that the Alteons cannot keep up with 200 requests per Alteon at 120 Hz for these small event fragment sizes (1400 Bytes).

VII. Summary

Hardware emulators based on an FPGA FE tester and a customized Alteon NIC have been studied to determine the limits of their performance. Understanding these limits allows the intelligent use of these emulators in a large-scale test bed for the ATLAS Dataflow network.

VIII. Acknowledgements

The authors want to thank the ATLAS T/DAQ group for its support for the research reported here.

IX. References

[1] Dobinson et al., "ATLAS TDAQ: A Network-based Architecture," ATLAS Document DC-059.
[2] Beck at al., "The base-line DataFlow system of the ATLAS Trigger & DAQ", contribution to this conference.
[3] H.P. Beck and F.W. Wickens, "The Message Format used by DataCollection..", ATLAS document DC-022
[4] Dobinson et al., "Testing and modeling Ethernet switches and networks for use in ATLAS high-level triggers", IEEE Trans. Nucl. Sci vol. 48, June 2001, p. 607.
[5] Barnes et al., "Testing ethernet networks for the ATLAS data collection system", IEEE Trans. Nucl. Sci. vol. 49, April 2002, p. 516
[6] Bee et al., "The raw event format in the ATLAS Trigger & DAQ, ATLAS Internal Note ATL-DAQ-98-129, Oct. 2002
[7] S. Stancu and M. Ciobotaru, "The use of Ethernet in the DataFlow of the ATLAS Trigger & DAQ", to be published in the proceedings of CHEP 2003