A RobIn Prototype for a PCI-Bus based Atlas Readout-System

A. Kugel, R. Männer, M. Müller, M. Yu
University of Mannheim, B6, 23-29, 68131 Mannheim
mmueller@ti.uni-mannheim.de

B. Gorini, M. Joos, J. Petersen
CERN, Geneva

B. Green
Royal Holloway, University of London

G. Kieft
NIKHEF, Amsterdam

Abstract

The Atlas RobIn Prototype is an FPGA and Processor based PCI device, which receives event data fragments on two HOLA SLink ports, buffers and delivers or deletes them on request. Thus it is one of the core devices of the Atlas Readout System (ROS). To study various PCI-Bus and Ethernet based ROS implementations the board offers two interface flavours for data requests: PCI and Gigabit Ethernet.

This paper presents a study of a ROS System based on an ordinary PC, hosting up to 3 RobIn Prototypes using the PCI-Bus for data exchange. The multithreaded host software uses an asynchronous messaging scheme optimizing the communication with the RobIns. First measurements indicate that a PCI-Bus based ROS can satisfy Atlas Data-Flow needs.

I. Introduction

The Atlas Readout Sub System (ROS) provides data access to events, which have been accepted by the first selection stage of the Atlas DAQ system (Level 1 trigger), to computing farms for further analysis. Event data arrives with a frequency of up to 75 kHz [1] on 1600 Readout Links (ROL) form the Atlas Readout Drivers (ROD). The ROS buffers this data in real-time; a volume of up to 160 MB/s per ROL link.

These requirements put high demand on the input of the ROS system. Data has to be taken on the input port for long periods without interruption. The data source, the Readout Driver (ROD), has insufficient capacity to buffer event data if the ROS is unable to take it.

On the ROS output side, buffered data is requested for further analysis by the Atlas Level 2 Farm. The latter rejects irrelevant events, and thus reduces the event rate down to a few kHz. To keep the analysis effort low, Regions of Interest (RoI) are defined for each event. This mechanism reduces the amount of data, requested for Level 2 analysis per ROL, to only about 1% [1] [2] of the input data. If the Level 2 Farm accepts the event, the ROS is requested to send the complete event data to the Event Filter Farm, which is the last data selection stage. This happens for about 3% [1] [2] of all incoming events. All other events get deleted inside the ROS buffer. The communication between ROS, Level 2 and Event Filter (requests, decisions and event data) runs over a Gigabit Ethernet Switching Network.

The high demands on the ROS input, and the use of a special link (SLink), necessitate a custom electronics device inside the ROS. This Readout Buffer Input (RobIn) is one of the main components of the Readout Sub System. Its task is to receive and buffer event data arriving via SLink from the RODs. Since the data request rate is much less than the input rate, several ROLs can be handled on one RobIn device. This concentration also has the advantage of economy. The Atlas Base Line architecture[1] [2] specifies four ROLs per RobIn board.

This paper presents a prototype RobIn to study two different ROS implementations based on PCI-Bus and Gigabit Ethernet. Furthermore the main focus is put on studying a complete PCI-Bus based ROS System including test results. This provides the first performance estimations of a PCI-Bus based ROS following the Atlas Baseline architecture.

II. ROS Hardware

A. The RobIn Prototype

The RobIn is the key component of the Atlas Readout System. It receives and buffers detector event data fragments with a size of up to 1.6 kB arriving with a frequency of up to 75 kHz.

Figure 1 shows the RobIn Prototype. It is a custom PC format PCI card, based on an FPGA and a microcontroller. Two[2] optical HOLA[3] SLink input ports connect the device to two ROD Readout Links. SLink protocol engine and input buffer management are implemented inside the Xilinx

[1] It is planned to later upgrade to 100kHz

[2] This is different to the Atlas Baseline proposal, and a result of the limited space on board

[3] HOLA: High Speed Optical Link for Atlas

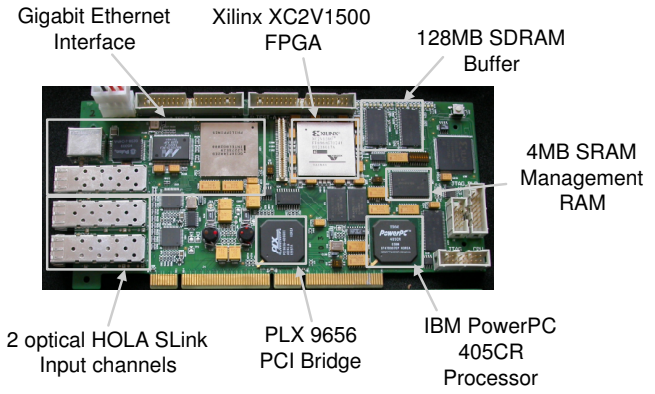

Figure 1: The RobIn Prototype. The main modules have been highlighted.

Virtex II FPGA (XC2C1500). The FPGA bursts incoming data directly into a pre-selected buffer page. The free page is provided by the IBM PowerPC 405CR microcontroller through a Free-Page-FIFO. The microcontroller has to avoid this FIFO becoming empty, so as not to stall the input stream.

128 MB SDRAM is available to buffer event data, 64 MB per ROL input. For buffer management purposes a small 2 MB SRAM is intended to take a used page hash table. This is used for localizing event fragments on data or clear requests.

The main buffer management is executed in the PowerPC processor. It provides empty pages to the SLink input engine inside the FPGA, stores the used pages against the event ID, and locates requested data.

Data requests or clear requests can be sent to the RobIn prototype via PCI-Bus or the Gigabit Ethernet Interface. In both cases the FPGA buffers the incoming messages and provides the PowerPC access to them. Message interpretation, execution and response preparation (if any) is done by the microcontroller. Response messages are sent by a DMA engine inside the FPGA using the selected interface (Ethernet or PCI). Event data fragments are gathered by this engine and added to the return message.

Finally a PLX9696 PCI bridge connects the prototype RobIn to the 64 bit, 66 MHz PCI-Bus. This device gates all RobIn accesses and provides Master DMA capabilities. The FPGA is able to access the PLX9656 through a 32 bit, 66 MHz local Bus. Alternatively an optical or electrical Gigabit Ethernet port is available.

A more detailed description of the prototype RobIn can be found in [3] and [4].

B. The ROS-PC

Several RobIn Prototype boards are hosted in an "off-the-shelf" PC. If the PCI-Bus is used for RobIn communication, a PC System with several bus segments provides best PCI bandwidth. Currently three Intel XEON systems are available for testing. Their processors frequencies vary between 2 GHz and 3 GHz. Each of these PCs has four PCI- buses and six available PCI slots. Figure 2 shows a ROS-PC setup as it is proposed in the Atlas Baseline architecture.

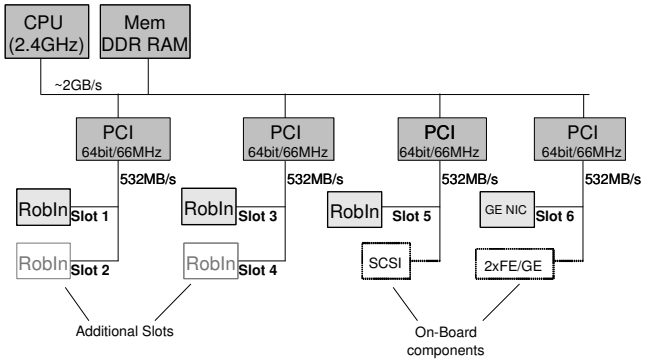

Figure 2: Schematic drawing of the PCI-Bus configuration within the ROS-PC. Three slots are used by RobIns and at least one Gigabit Ethernet card uses another slot. Open slots may carry additional RobIns. At least one Fast Ethernet adapter is located on the PC mainboard.

Three of the four available PCI segments are occupied by three RobIn Prototype boards. The remaining segment holds at least one Gigabit Ethernet card. Two more Ethernet adapters are located directly on the PC mainboard. Having four ROLs handled by one RobIn, the complete ROS-PC is able to receive data from 12 Readout Links. In this setup, the full PCI bandwidth is available to each component, the three RobIns and the Ethernet adapters. Using all available slots allows adding up to six RobIns and 24 ROLs into a single PC.

III. ROS Software

To mediate between requests of the Level 2 and Event Filter Farm, and the RobIns, the ROS-PC runs a multithreaded software written in C++ on top of the Linux Operating System. It:

- initializes and controls the RobIn boards,
- listens for data and clear requests on Ethernet,
- forwards data and clear requests to the RobIn boards,
- gathers the event data fragments from the RobIns, formats them and sends them to the requestor.

Figure 3 shows a functional diagram of the ROS-PC software. Incoming messages are received by the input handler (Trigger), which runs in a separate thread. It puts all incoming messages into a thread-safe request queue for further processing. A configurable number of Request Handlers are responsible for execution of the request messages. Each Handler runs in a separate thread. To process the messages inside the request queue, a free Request handler thread is selected. It reads the request message out of the queue and accesses the RobIn hardware using the abstract Fragment Manager interface. Each ROL on each RobIn has one dedicated Fragment Manager instance in the ROS-PC application. In case of a data request, returning event fragments are collected and sent by the Gigabit Ethernet interface to Level 2 or the Event Filter.

The detailed communication between the ROS software and the RobIn is completely abstracted by the Fragment-

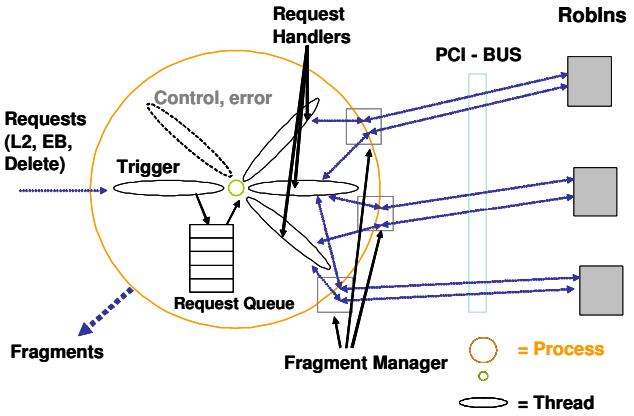

Figure 3: Multithreaded organization of the ROS PCs software. Incoming requests are queued by a Trigger thread. Several request handlers execute the requests by accessing the RobIns through the Fragment Manager interface.

Manager object. The mechanism has been optimized to speed up data requests. It is derived from earlier prototypes [5].

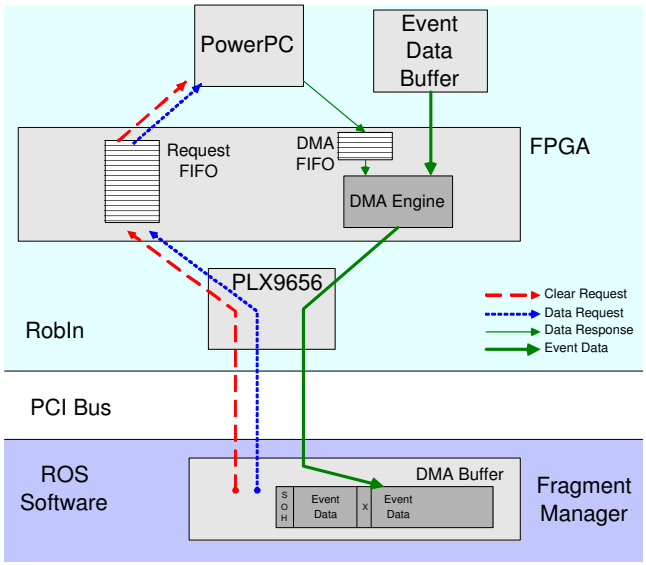

Figure 4: The Fragment Managers communication with the RobIn. Clear requests pass the RobIns FPGA and are executed by the PowerPC with no response. Requests are handled and answered by the PowerPC using the FPGA DMA engine.

Figure 4 shows the request and data flow between the ROS- PC (the Fragment Manager) and a RobIn PCI board. RobIn Requests are sent by single PCI writes (for small messages, e.g. event data requests), or by a Bus Master DMA (for large messages, e.g. clear requests), to a FIFO inside the FPGA of the RobIn. All RobIn PCI addresses are memory-mapped [4] into the address space of the Host-PC.

The available request messages to the RobIn and their format is described in [6]. An event data request message consists of three 32bit words plus the event ID. A clear message consists of four 32bit words plus a large number of event IDs (usually 100).

The PowerPC processor of the RobIn polls on the FPGA's input FIFO and reads the message. In case of a clear message, it executes it by removing the matching events from the buffer. In case of a data request, it forms a response which instructs the FPGA's DMA engine to send event data from the RobIn buffer directly, via the PCI-bus, to the Host's DMA buffer.

When writing event data, the RobIn is able to randomly access the DMA Buffer inside the Host-PC. This allows to write event data to a pre-defined position and prevents unnecessary memcopies of the ROS Software.

To signal the end of an event data transfer, the RobIn delays the first 32bit word of the event fragment data. On the Host PC, the Fragment Manager instance clears the first word of the destination DMA buffer prior to the request to zero. When the RobIn finalizes the data transmission it sends the first word of the event data fragment. This is detected by the Fragment Manager software by polling on the first word of the destination DMA memory. Since polling blocks the CPU, the thread forces the Operating System to re-schedule all threads. Other threads may now use the CPU, while the polling thread waits for data arrival. This prevents a high loss of performance due to extensive polling of the Request Handler threads.

IV. Measurements and Discussion

Several tests have been performed, to characterize the ROS Software and the PCI-Bus utilization. Since the RobIn prototype was not available from the beginning, a similar PCI board, the MPRACE[7], has been used to emulate PCI-Bus messaging. MPRACE is composed of the same key components: a Xilinx Virtex II FPGA and the PLX9656 PCI bridge. Only the PowerPC processor is absent. Its tasks are executed within the FPGA. Event fragment data is pre-loaded into the MPRACE RobIn emulator; no SLink has been used during the tests.

All Measurements have been done using one of the three PCs mentioned in section II-B, equipped with up to three MPRACE RobIn emulators. To estimate results with four ROLs per RobIn, the ROS Software is able to instantiate four Fragment Managers per MPRACE. Measurements with I/O to Gigabit Ethernet have been done using a second Xeon System, running a request emulator.

A. PCI Data Request and Clear Performance

Diagram 5 shows the data request performance depending on the number of active ROS software Request Handler threads. A huge number of data requests are generated internally, and distributed to the available threads. The requests are passed to a single MPRACE RobIn board. Returning data is sent to an output emulator, which does no further operation. Finally the number of successful requests per time (the request rate) is reported.

In case of only one Request Handler, data fragments are processed sequentially by the ROS PC software and the

[4]Memory-mapping allows to directly write to an address of the Host-PC which leads to single cycle reads or writes on the PCI-Bus

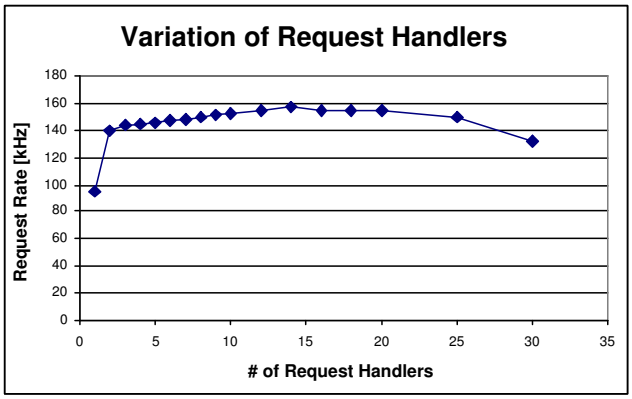

Figure 5: RobIn data request rate depending on the number of Request Handler threads measured with the one MPRACE board.

RobIn. This implies waiting periods, which can't be used for other activities. The result is a very low rate request rate.

Increasing the number of threads increases the request rate too, because other threads may process events while one thread is waiting for the RobIn. A maximum is reached when having 14 threads. Having more increases the overhead for task switching, which lowers the request rate again.

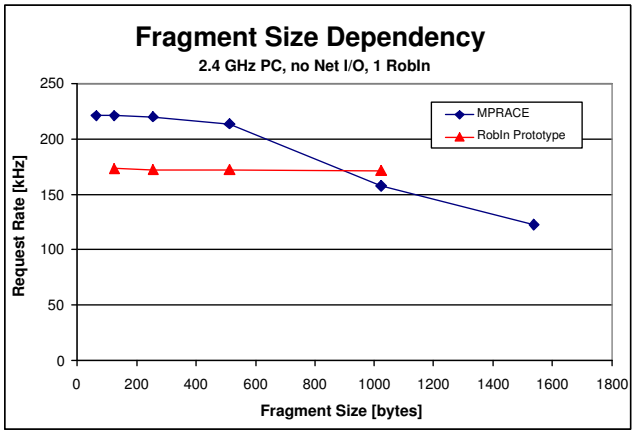

Figure 6: The dependency of the data request rate depending on the Event Fragment size measured with the MPRACE RobIn emulator and the real RobIn Prototype hardware. The number of Request Handler threads was set to 8

When varying the fragment size, the request rate decreases for event fragments larger then 512 bytes. For smaller fragments the request rate stays constant. This is shown in diagram 6. In case of small fragments, fixed overheads (memory allocation, RobIn request message decoding, ...) overlap the very short PCI-Bus data transmission time. Thus, up to 512 bytes only these fixed overheads determine the RobIn request rate. In case of large fragments the PCI-Bus data transmission time plays a more important role and limits the request rate. A decrease of the request rate with increasing fragment sizes can be observed.

Diagram 6 includes the measured request rate of the RobIn prototype board, which can be compared with the MPRACE results. It has been observed that the results with the RobIn prototype are in the same order of magnitude. This approves the MPRACE emulating the RobIn prototype for the first time.

To finalize the examination of the basic RobIn accesses, the execution time of clear requests has been measured. This is plotted in diagram 7 depending on the number of Event IDs carried by the clear message towards the RobIn. For up to 100 IDs, the measured time increases linear up to 13 μs. A linear fit produces a fix offset of 6.8 μs. This offset is the sum of the time to prepare the clear message, set up the Bus-Master DMA on the PLX PCI-Bridge and to decode the message on the MPRACE. Without this overhead, the time to delete is 0.06 μ per Event ID.

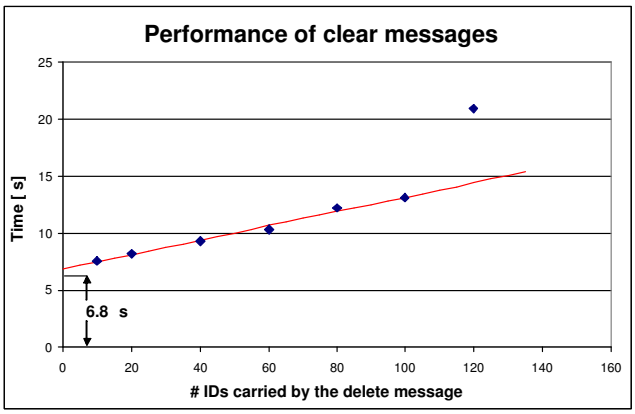

Figure 7: The time to send a clear message to the RobIn and execute it. Varied is the number of Event IDs carried by the clear message.

B. *Effect of Network I/O*

Adding real I/O to Level 2 and the EF switching network reduces the overall ROS performance due to the additional CPU load for handling the output to Gigabit Ethernet. This has been measured and is drawn in diagram 8. It shows the maximum sustainable Level 1 input rate to the RobIns depending on the Level 2 accept rate, while 2% of the incoming event fragments are requested by the Level 2 trigger (per RobIn). The maximum Level 1 rate has been back-calculated from the RobIn's request rate.

The test setup for the measurement in diagram 8 consists of two PCs. One ROS-PC with 3 MPRACE RobIn emulators and another PC, emulating requests from the Level 2 and the Event Filter farms. Each MPRACE RobIn emulator represents a RobIn with four input links from the Readout Drivers. Thus the ROS-PC handles 12 ROLs, which matches the Atlas baseline architecture.

Diagram 8 shows that including I/O traffic on the Gigabit Ethernet output, reduces the performance of the full scale ROS-PC by a factor of 3. But only for high Level 2 accept rates (large traffic to the Event Filter), a significant traffic is generated on Gigabit Ethernet; about 70 MB/s with 8% Level 2 accept fraction.

For small Level 2 accept fractions the limiting factor can

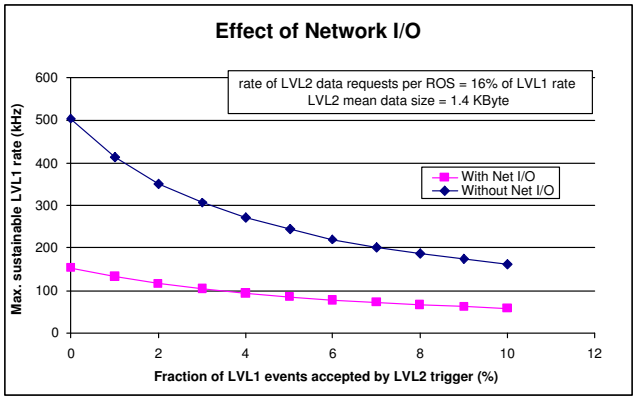

Figure 8: Maximum sustainable Level 1 rate of a ROS System handling 12 ROLS on 3 MPRACE RobIn boards, 4 ROLs per RobIn. The measurements have been done with and without I/O to the LVL2 and EF switching network on a 2 GHz PC

be identified to be the ROS-PCs CPU power. This is shown in more detail in the next sub-section.

C. Scaling with the Host CPU frequency

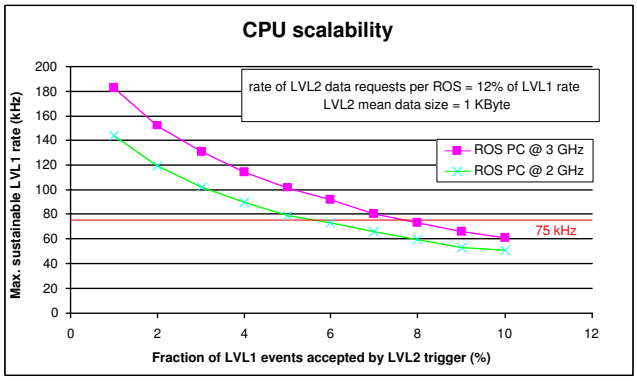

Figure 9: Maximum sustainable Level 1 rate of a ROS System handling 12 ROLS on 3 MPRACE RobIn boards. Measurement has been done with I/O to the LVL2 and EF switching network on a 2 GHz and 3 GHz PC

Diagram 9 shows the performance of a full scale ROS-PC, tested on two different PCs, with 2 GHz and a 3 GHz processors. Tests are again done with the 3 MPRACE boards emulating input form 4 Readout links.

Switching to the faster CPU increases the overall performance by 25% for small Level 2 accept rates and by 16% for large accept rates. This gives a clear indication that the ROS-PC is limited by its computing power for low Level 2 accept rates. For high Level 2 accept rates the improvement is less, which indicates that ROS System performance starts to depend on the Gigabit Ethernet line speed too. Having a Level 2 accept rate already generates a traffic of about 80 MB/s and the theoretical limit for Gigabit Ethernet is 125 MB/s.

Finally this measurement approves that a ROS System, based even on the slow PC, is able to fulfil the required Atlas Level 1 input rate of 75 kHz (with 1% Level 2 ROI request fraction and 3% accept fraction).

V. Conclusions

The presented PCI-Bus based ROS enables to concentrate at least 12 Readout Links to one Network output to the Level 2 and Event Filter Switching networks. First measurements of a complete system, comprising PCI-Bus messaging performance and I/O to Gigabit Ethernet, show that the Atlas requirements can be fulfilled. The measurements have been done with hardware emulating the RobIn PCI messaging. Thus the results have to be approved with the real RobIn prototype hardware. First tests have already been done and show that the RobIn's performance is in the same order of magnitude.

Detailed measurements have shown that the performance of the PCI-Bus based ROS is currently limited mostly by the speed of the Host-PC CPU. Significant traffic on Gigabit Ethernet is generated only in case of large Level 2 accept rates, but this is still not close to Gigabit Ethernet line speed.

Further tests with the prototype RobIn may also include real input via the HOLA SLink ports. This proves the presented Level 1 input rates, witch have been back-calculated from the measured PCI-Bus request rate.

References

[1] J. C. Vermeulen et al, The Baseline DataFlow System of the ATLAS Trigger & DAQ, *LECC 2003, Amsterdam*, Oct. 2003.

[2] ATLAS HLT/DAQ/DCS Group, ATLAS High-Level Trigger, Data Acquisition and Controls Technical Design Report, CERN, July 2003.

[3] B. Green, G. Kieft, and A. Kugel, ATLAS TDAQ/DCS ROS Prototype-RobIn HLDD, *ATL-DQ-EN-0001*, Sept. 2002.

[4] B. Green, G. Kieft, and A. Kugel, ATLAS TDAQ/DCS ROS Prototype-RobIn DLDD, *ATL-DQ-EN-0002*, Sept. 2002.

[5] R. Bock, J. A. Bogaerts, P. Werner, A. Kugel, R. Männer, and M. Müller, The Active Rob Complex: An SMP-PC and FPGA based solution for the Atlas Readout System, *IEEE Realtime 2001 Conference*, pp. 199 - 203, June 2001.

[6] B. Green, G. Kieft, and A. Kugel, ATLAS TDAQ/DCS ROS Protoype-RobIn Software Interface, *ATL-DQ-EN-0003*, Sept. 2002.

[7] The MPRACE FPGA Co-Processor, http://www-li5.ti.uni-mannheim.de/fpga/index.shtml