ATLAS Trigger/DAQ RobIn Prototype

2003 IEEE Real Time Conference

B. Green, G. Kieft, A. Kugel, M. Mueller, M. Yu

Abstract—The *ATLAS* Trigger/DAQ (TDAQ) system connects via 1600 Read-Out-Links (ROL) to the *ATLAS* sub-detectors. Each Read-Out-Buffer (RobIn) prototype attaches to 2 ROLs, buffers the incoming event data stream of 160MB/s each and provides samples upon request to the TDAQ system. We present the design of the PCI-based RobIn module, which is built around a XILINX XV2V1500 Field-Programmable-Gate-Array (FPGA), together with initial results from rapid prototyping studies.

I. INTRODUCTION

THIS document describes the design and implementation of the *ATLAS*¹ RobIn (Read-Out-Buffer) prototype, which is a part of the *ATLAS* Trigger/DAQ (TDAQ) baseline architecture, as documented in the (draft) TDAQ Technical Design Report (TDR) [5].

Section II describes the *ATLAS* TDAQ architecture and the relevant options which still need to be investigated. Section III provides the requirements and describes functionality and the high-level design of the RobIn. Subsequently section IV shows the most important aspects of the hardware and the software implementations. The overall project status is presented in section V, together with initial results obtained during rapid-prototyping studies. Finally, section VI displays the plans and schedule guiding the future work.

II. *ATLAS* TDAQ ARCHITECTURE

In *ATLAS*, data from the Read-Out-Drivers (ROD) of individual sub-detectors are transmitted to the TDAQ system via optical point-to-point Read-Out-Links (ROL). The total number of ROLs is in the order of 1600, the individual bandwidth is up to 160MB/s at the maximum event rate of 75 kHz, generated by the first-level-trigger (L1). On the receiving side – the Read-Out-System (ROS) – data are

accepted initially by RobIn modules and stored into one buffer per ROL. Event data from the RobIn will be requested by the second-level trigger (L2) and the event-builder network (EB). The L2 system operates at the full L1-rate of 75 kHz but uses locality information – the regions of interest (RoI) – which limits the amount of requested data to 2% of the detector volume per event. This leads to an average request rate per RobIn related to L2 traffic of 3 kHz. The EB operates at the L2 accept rate, which is again in the order of 3 kHz. The total output load per RobIn, both in terms of rate and bandwidth, is therefore somewhat below 10% of the input load. The DataFlow (DF) [1] subsystem is responsible for the efficient transport of requests and data between the components of L2, EB and ROS.

The architecture of the ROS is not finalized at present rather there is a baseline architecture plus an option, both of which will be extensively studied in the near future.

In the baseline ROS each RobIn is realized as a PCI card serving 4 ROLs. 3 of these PCI cards are housed in a ROS-PC with multiple PCI buses. Data (and Delete) requests from L2 and EB arrive over the Gigabit Ethernet network (GbE) at the ROS-PC and are forwarded to the appropriate RobIn. In turn, event fragments from the RobIns are collected over PCI by the ROS-PC and sent out via GbE. Measurements [4] with RobIn-Emulators have shown that this architecture is able to provide the required performance.

In the alternative approach every RobIn is directly connected to the main GbE network via a private network interface. As a consequence, the PCI bus does not carry event data any more. This architecture provides a better scaling and allows for a more compact packaging of the RobIns (e.g. industrial PC with a large number of PCI slots). However there is some trade-off as well, in terms of required network ports and higher load on some parts of the system due to the increased number of data sources.

III. ROBIn DESIGN AND APPROACH

The RobIn prototype, which is the subject of this paper, is especially designed to allow investigations of the full range of potential scenarios, from the standard PCI-alone operation to the extreme of a stand-alone network based environment, without any host computer at all. In addition, the modular design allows a very smooth transition to any final RobIn with 1 to 4 ROLs.

Manuscript submitted on May 30, 2003.

B. Green is within the Physics dept. of the Royal Holloway University of London, UK. (e-mail: B.Green@rhul.ac.uk)

G. Kieft is within the Electronics dept. of the National Institute for Nuclear and High-Energy Physics, Amsterdam, NL. (e-mail: n55@nikhef.nl)

A. Kugel is within the dept. for Computer Science V of the University of Mannheim, Germany. (e-mail: kugel@ti.uni-mannheim.de).

M. Mueller is within the dept. for Computer Science V of the University of Mannheim, Germany. (e-mail: mmueller@ti.uni-mannheim.de).

M. Yu is within the dept. for Computer Science V of the University of Mannheim, Germany. (e-mail: yu@ti.uni-mannheim.de)

¹ *ATLAS* is one of the high-energy-physics (HEP) experiments at the Large-Hadron-Collider (LHC) which will start operation in 2007.

A. Requirements

The main functionality of a RobIn in the TDAQ system described previously is to **RECEIVE, BUFFER, DELIVER** and **RELEASE** the event fragments arriving from the detector. In addition, the following requirements must be satisfied:

1. Dual S-LINK [11] Input, max. 160MB/s per link, XOFF capability.
2. Local buffer to compensate L2 trigger latency of 10ms plus good safety margin.
3. Efficient buffer management, capable of sustained event rate of 75 kHz.
4. PCI and GbE Interfaces for control and data requests at a rate of 3 kHz for L2 and EB each and 1kB average event size.

B. High-Level Design

The design process was started in March 2002 with the aim to provide a single *ATLAS* prototype RobIn replacing the variety of RobIns in use at that time. The new design is based upon the previous prototypes, where useful, whilst avoiding the bottlenecks. A total of 4 RobIns were considered in this design analysis [6][9]: MFCC-RobIn, I960-RobIn, FPGA-RobIn and SHARC-RobIn. They all provide the main RobIn functionality and exhibit a similar architecture: a single S-Link is attached to a programmable logic device (e.g. FPGA), data from the S-Link is temporarily stored in a local buffer, and data are transferred via a PCI bridge device to the host and are deleted upon request later on.

Fig. 1: Block Diagram

The block diagram of the RobIn prototype is shown in Fig. 1. The processor (CPU) is completely detached from the main data path, which avoids the main bottlenecks experienced with older RobIns. It performs the book-keeping using its private RAM, as well as request handling and all low-rate operations like monitoring. The central FPGA provides all the required connectivity and is fully reconfigurable to realize the intended scenarios. The input links and the buffers (one per link) attach directly to the FPGA. The TDAQ interface is realized with a dedicate PCI bridge and with a dedicated MAC/PHY chipset

The design of the RobIn prototype is documented in a number of design documents [6][7][8] and has been

approved during a technical review [3] in October 2002 at CERN.

IV. ROBIn IMPLEMENTATION

The core of the RobIn prototype is made from two devices, a XILINX XC2V1500 FPGA and an IBM PowerPC405. The FPGA covers all high performance, time critical functions like input link protocol handling and buffer memory access. Per ROL 64MB² of SD-RAM are available to buffer event fragments. The processor handles the requests from the DF subsystem and performs any kind of book-keeping. The processor has additional 64MB of SD-RAM. The management of fragments and buffers is shared between FPGA and processor in way that the processor never needs to access data in the buffer. This approach avoids the various bandwidth problems observed with some older prototypes. The PCI interface is provided by a commercial 64bit/66MHz PCI bridge (PLX PCI9656) which is well-known from another FPGA-card (MPRACE [14]) developed at Mannheim University. The advantage from this approach is three-fold:

1. At the hardware level the interface between FPGA and host PC can be re-used.
2. The low-level driver software can be re-used.
3. The ROS software interfacing to the RobIn could be developed and tested in advance.

For the network interface a discrete Gigabit MAC was chosen mainly because the XILINX Gigabit core was not considered to be mature. The Intel IXF1002 device provides two completely independent FIFO³-like transmit and receive interfaces which require little control effort. For the physical media attachment the PHY device MARVELL 88E1011S was selected, which provides both electrical and optical media with an auto-detect feature. The mechanical part of the S-LINK specification doesn't allow to house more than one link on a PCI board. To support multiple ROLs the components for 2 channels of the 2.5Gbit/s optical S-LINK (HOLA [10]) are integrated on the PCB. A complex programmable logic device (CPLD) provides basic board control functionality (e.g. Resets, JTAG support, etc.). FLASH memory is available for the processor boot code and FPGA configuration.

The FPGA attaches to the processor's external bus, which runs at 66MHz, with 32 bit non-multiplexed operation. This clock is as well used as the main system clock inside the FPGA and for it's interconnects to PCI bridge and MAC. The buffers operate from a 100MHz clock which is generated inside the FPGA. The ROLs need a 125MHz transmit clock which is common for the 2 links. In addition each ROL provides a 125MHz receive clock. The decoupling of the 5 clock domains is done in the FPGA by using asynchronous FIFOs.

² A 64MB buffer allows 400ms of latency at 160MB/s input bandwidth.

³ FIFO: First-In First-Out is equivalent to a queue. FIFOs are hardware components, which can be realised in the FPGA.

A. Buffer Management

Buffer management is THE central function of the RobIn and a large amount of experience has been gained in this area in the previous prototype work. The design presented is based on the paged buffer management scheme developed by the UK group for the I960-RobIn, with some optimizations applied.

The buffer memory is logically divided into a number of pages, which are used to store the event fragments. The page size is 1kB – which is equivalent to the average fragment size – but can be arbitrarily adjusted. Larger fragments consume multiple pages. Upon initialization all available pages – with the given memory layout we have 64k pages of 1k size – are entered into the stack of free pages. The buffer management is performed at two levels, one operating at the word-level (BufMgr-IN) and the other one operating at the page-level (BufMgr-OUT). BufMgr-IN runs on the FPGA whereas BufMgr-OUT runs on the CPU. All communication between the 2 levels is done via a set of FIFOs which communicate information about free and used pages.

Fig. 2: Main Flow of Data and Control

Fig. 2 shows a detailed view of the building blocks which implement this functionality. A ROL is attached via the external ROL-IF (optical transceiver and SerDes) to a ROL-Handler, which runs the S-LINK destination (LDC) protocol engine. The internal S-LINK interface is then connected to the INPUT-Handler which decodes the fragments sent from the ROD – e.g. indicates the field of the event number (L1ID) – and performs low-level error checking. The BufMgr-IN receives the complete fragment with the word-by-word decoded header identifiers and error markers. At the start of a new fragment it takes a new value from the free-page-FIFO (FPF) and translates it into a starting address in the buffer. Next, it starts to write the entire fragment into the buffer. For longer fragments it obtains new pages as required. For every page it uses to store fragment is write a fragment-information-entry (FIE) into the used-page-FIFO (UPF). The FIE contains the page number, the L1ID plus additional header information, the number of words in the page and a status field with end-of-fragment and error markers. At this level fragments are never dropped due to

errors. If there are no more entries in the FPF an XOFF command is sent down the ROL, followed by an XON after new pages have become available. Both FPF and UPF operate at the full L1-rate of 75 kHz and have a minimum depth of 100 entries. After a data request the DMA engine picks up the fragment from the buffer and sends it to the DF subsystem. The BUFFER-ARBITER controls the memory access by switching between BufMgr-IN and DMA, allowing for a minimum burst size of 64 bytes. At 100MHz SD-RAM frequency this guarantees a sustained bandwidth of 160MB/s for the fragment input and approx. 100MB/s for the DMA engine.

The task of the processor is to do the book-keeping of all event fragments in the buffer. A fragment is identified by its 32 bit L1ID and in general⁴ the L1ID is monotonously incrementing. To find quickly the information related to a L1ID a simple hashing algorithm is used:

1. A hash-key is created from some lower bits of the L1ID
2. All pages with the same key are stored in a linked list

The time to find a fragment is determined by the average search depth in the linked list. The large amount of management memory available to the processor enables to use a large 20 bit hash-key which results in an average search depth of 1, provided L1IDs are sequential.

When reading FIE entries from the UPF the processor checks the status flags. There are a number of errors which will normally lead to discard the page immediately, e.g. incorrect format. However for debugging purposes the processor may decide to keep erroneous entries as well. If a duplicate L1ID is recognized, the pages allocated for the previous fragment are freed and the new fragment is entered at a new location. The new fragment is marked as a “duplicate”. Finally the processor periodically moves free pages from the stack to the FPF.

The memory required for the processor to perform the buffer-management is approx. 4MB per ROL, which can be calculated as follows:

The number of pages is fixed to be 64k⁵, requiring a 16 bit page index which can be represented by a short-int value. The stack of free pages is realized as a static array of 64k shorts. Another static array is used to hold 64k management entries of 28 bytes each, which consist of 2 short pointers (previous, next) for the linked list, the FIE entry and one 32 bit word for the total fragment size. Using a 20 bit hash key the hash-array is a static array of 1M shorts, each pointing to the first page entry in the corresponding list. In terms of bandwidth the processor’s data memory is accessed at approx. 8MB/s and the FPGA at approx 2MB/s for UPF and FPF.

⁴ In special cases the L1ID is not monotonous. In addition a unique L1ID cannot be guaranteed.

⁵ 64k pages is 64MB and 1k page-size. A smaller page-size is not foreseen.

B. Request Handling

Upon a data request from DF the processors look for the fragment and transfers address and transfer size page by page to the DMA engine. Upon a release request the pages used by the event are removed from the linked list and returned to the stack of free pages, from which they will be subsequently moved to the FPF. The expected rate for the data requests is in the order of 6 kHz, which leads to an average output bandwidth of 6MB/s. Clearly the rate of release requests must match the L1-rate in order to prevent buffer overflow. In order to lower the message rate on the DF-port the release requests are normally distributed in groups of 100.

Although the RobIn provides 2 different interfaces to DF, via PCI or via the network, the main operation is completely independent from the type of DF-interface. The RobIn is required to place a ROB-header [12] in front of the raw data from the ROL, when sending requested fragment data to DF.

All interaction is based upon messages and the generic formats of RobIn messages are detailed in [8]. In general, a message shall specify its purpose, indicate the address to which a reply shall be sent and contain information specific to the purpose of the message. In the PCI environment the messages sent over PCI are identical to the generic RobIn messages. Request from PCI enter the Request-Handler via a FIFO (MsgFifo). Next, the processor picks-up one message at a time and checks if it's a valid request. It prepares a so-called media-header structure from the indicated destination address and the media information, which is *PCI* in this case. Then the request with the media-header is forwarded for execution to the appropriate control entity, which is one of Control, Monitor-Mgr or BufMgr-OUT. If an explicit response is expected, it will normally consist of information prepared by the processor (e.g. statistics counters) plus data from the buffer, in case of a fragment request. In order to send the response to DF, the RobIn formats a data block consisting of the media-header, an optional user-header (e.g. the ROB-header or statistics) and information about data from the buffer (address and size). This block is written into another FIFO (HdrFifo). From there it is processed by the DMA engine, which assembles the final message and sends it to the host via the PCI bridge via the direct-master interface of the PCI bridge. The detailed procedure is as follows:

Upon application start the host allocates a number of physically contiguous memory blocks per RobIn and builds up a memory pool. For each request one block is taken from the pool and used as the destination address. The host clears the first word (offset 0) of the header section of the block prior to using it for a request. The RobIn will return the ROB-fragment (or ROB-response) to that address, with initially setting offset 0 to the value 1, indicating active processing. After all data has been written, the RobIn overwrites offset 0 with the Start-Of-Header marker, indicating completion. By monitoring the value at offset 0 the host is able to detect the status of the processing.

In the network environment the approach is almost the same. At the network layer the message format is defined in

[13] and the chosen protocol is raw-Ethernet. The network message is translated in the network adaptation layer to the appropriate RobIn message and the media-header structure is initialized with the return address and some other information. The subsequent processing is identical to the previous PCI case. Only if a fragment consists of multiple pages then BufMgr-OUT needs to update a sequence counter in the media-header for every page, according to the multi-frame message specification in [13]. The required bandwidth for the processor to read and write the message and header FIFOs is in the order of 2MB/s each.

V. STATUS AND RESULTS

10 RobIn boards (Fig. 3) are available since March 2003. All hardware components have been verified to work properly. Most parts of the FPGA design have been tested individually:

1. Processor access
2. PCI access
3. Network interface
4. ROL operation
5. Buffer access
6. DMA operation

Fig. 3: RobIn

The buffer-management on the processor is ready for test, but might still need some optimization.

The PCI communication mechanism has been prototyped on MPRACE with very promising results [4]: With a single PCI board a request rate of 150 kHz can be sustained, at 1kB fragment size. When emulating a 12 ROL system with 3 MPRACE boards the performance required for ATLAS can be achieved (75 kHz L1 rate, 2 % L2 rate, 3 kHz EB rate).

Concerning tests in the PCI environment there is only little work left, apart from integrating the various VHDL modules which build-up the FPGA design. For network tests the basic input and output routines are available but the raw-socket protocol still needs to be implemented. In addition there is extended initialization software required, e.g. to download the mapping of DF node identifiers to Ethernet addresses.

VI. FUTURE WORK

For the near future the highest priority is to prepare a RobIn with the required basic functionality and to integrate it into at least 2 different scenarios, for PCI and network tests. Up to the end of 2003 the final design of the RobIn should be developed, based upon the results with this prototype. That

version will be produced in quantities in 2004 and early 2005. The most important milestones are given below:

1. June 03: RobIn Application and VHDL complete
2. Sept 03: Performance Testing and Optimization
3. Dec 03: Final RobIn design ready
4. May 04: Pre-series of final RobIn ready

VII. REFERENCES

- [1] The ATLAS TDAQ DataFlow community
<http://atlas.web.cern.ch/Atlas/GROUPS/DAQTRIG/dataflow.htm>
- [2]
- [3] Ph. Farthouat, R. Hauser, B. Martin, A. Racz: "FDR of the ROBIN prototype", CERN publication, Nov. 2002, ATLAS intranet
<https://edms.cern.ch/file/359014/1/robin-pdr.pdf>
- [4] B. Gorini, "Results of measurements on the ROS baseline architecture", *Transparencies at ATLAS TDAQ Review*, CERN, Mar. 2003, CERN intranet.
- [5] Atlas HLT/DAQ Technical Design Report (TDR), Draft, CERN, May 2003, <http://atlas-proj-hltDAQcs-tdr.web.cern.ch/atlas-proj-hltDAQcs-tdr/>
- [6] B. Green, A. Kugel, G. Kieft, "RobIn HLDD", CERN intranet, Sep. 2002, <https://edms.cern.ch/file/356324/2.4/hldd.pdf>
- [7] B. Green, A. Kugel, G. Kieft, "RobIn DLDD", CERN intranet, Sep. 2002, <https://edms.cern.ch/file/356328/2.3/dldd.pdf>
- [8] B. Green, A. Kugel, G. Kieft, "RobIn SWID", CERN intranet, Sep. 2002, <https://edms.cern.ch/file/356332/2.2/swid.pdf>
- [9] The ROS group: "Summary of Prototype RoBInS", CERN intranet, Feb. 2002, <https://edms.cern.ch/file/382933/1/RoBInSummary.pdf>
- [10] ATLAS Readout Link recommendation, CERN intranet, http://edms.cern.ch/file/332389/1/rod_rol.pdf
- [11] HOLA S-Link documentation, CERN intranet, <http://hsi.web.cern.ch/HSI/s-link/devices/hola>
- [12] Bee, D. Francis, L. Mapelli, R. McLaren, G. Mornacchi, J. Petersen, F. Wickens, "The raw event format in ATLAS Trigger & DAQ", *DAQ 98-129*, Version 2.2, CERN, 11th October 2002, <http://doc.cern.ch/archive/electronic/cern/others/atlnot/Note/daq/daq-98-129.pdf>
- [13] The Message Format used by DataCollection in the ATLAS TDAQ Integrated prototype: DC note, CERN 2002, <http://atlas.web.cern.ch/Atlas/GROUPS/DAQTRIG/DataFlow/DataCollection/docs/DC-022/DC-022.pdf>
- [14] A. Kugel, "MPRACE, preliminary documentation", CERN intranet, 2002, <http://akugel.home.cern.ch/akugel/mpRace/>