AUXbus: An Event Driven Bus-based DAQ Systems

A. Aloisio, S. Cavaliere, F. Cevenini, D. J. Fiore, L. Merola

Dipartimento di Scienze Fisiche, Università di Napoli "Federico II"
and I.N.F.N. Sezione di Napoli
Mostra d'Oltremare - Pad.20 80125 Napoli, Italy

Switch-based DAQ systems move the subevent assembling and data framing functionalities to the front-end readout crates. AUXbus is a bus-based protocol specifically developed to achieve true event driven behavior with no software overhead. It allows designing a high performance readout architecture matching the requirements imposed by a switch fabric used as an event builder.

Key word: Bus protocol, Event building, Backplane.

1 Introduction

AUXbus is a custom parallel bus protocol designed to provide optimum performances for DAQ systems in HEP experiments. AUXbus targets this specific application field by featuring event-driven operations, sparse-data scan capability and random length block transfer cycles up to 15 MTransfer/s. AUXbus also includes resources for extensive diagnostic checks and performance monitoring. Its ability to distribute flow control as well as synchronization and trigger signals, greatly simplifies the DAQ final design. AUXbus uses the lines available on the outer two rows of the VME/J2 connector (64 TTL lines). Precision timing pulses - as trigger signals - are distributed as differential ECL levels onto the JAUX bus, which has been redesigned from the original CERN specifications. A monolithic backplane which merges VMEbus (A32/D32), AUXbus and the modified JAUX subsystem has been developed. Careful layout design rules have been employed to achieve tight control on line impedance for optimum signal integrity. TDR technique has been intensively used to characterize the AUXbus backplane lines and to fine tune the PCB manufacturing.
The AUXbus architecture has been adopted by the KLOE experiment[1] at the DAΦNE Φ-factory in Frascati INFN National Laboratories, Italy. For this experiment we have also designed an AUXbus readout controller which is the building block of the KLOE DAQ System.[2]
In this paper we describe the bus protocol and the monolithic backplane's features. We also present preliminary test results of our architecture in the KLOE DAQ framework.

2 AUXbus protocol

The AUXbus protocol specifies the rules for transferring data across the bus, but it also demands that specific hardware guidelines are followed in designing both AUXbus slave and readout controller boards. In order to implement a true event driven architecture, AUXbus requires that both the DAQ slave boards and the controller share the same trigger pulses. When a trigger pulse is generated, each board in the system increments its own event number counter. Slave boards perform data conversion, label the acquired data with the respective event number, then push the frame into a data queue. In the same fashion, the controller stores the event number in a trigger queue and proceeds to readout the slave boards on an event-by-event basis, retrieving the event number to be processed from the trigger queue.

A fully compelled handshake is used to exchange data between DAQ boards and the AUXbus controller. Some of the key features are broadcast operations labeled by event numbers, block oriented data transactions and a logical and geographical addressing scheme which allows handling up to sixteen slave boards per crate. The AUXbus protocol has three phases: a sparse data scan, a data readout cycle and a synchronization-check cycle. An AUXbus transaction always starts with a sparse data scan. The event number is broadcast to all the slaves onto a trigger bus (XT[11:0]), strobed by the XTRGV line. Those boards with valid data belonging to that specific event, assert their individual flag (XSDSi). This maps the data availability into a parallel pattern, validated by a wired OR broadcast acknowledgement line (XBK). The AUXbus controller then establishes a connection with the responding slaves using an address strobe (XAS) and a 4-bit address bus (XADDR[3:0]), which gives the geographical location of the slave board to be accessed. The readout cycle then proceeds by means of random length block transfers. If no slaves with valid data were found during the sparse data scan phase, the controller skips the readout cycle and starts processing the next event number in the queue with a new broadcast.

AUXbus is fully asynchronous: the 20-bit data bus (XD[19:0]) is handled by a data strobe (XDS) - data acknowledgement (XDK), VME-like handshake. Different from the VMEbus, slave boards flag their last piece of data transferred using the End-of-Block line (XEOB). This terminates the connection and allows the controller to address the next board in the queue or to end the event number processing if no more boards with data remain.

For the AUXbus system to operate properly, each slave board and the controller as well, must contain the same value in the on-board event number counter. Thus, another feature of the protocol is the synchronization-check cycle. It is performed in the same manner as a readout cycle, except that instead of requesting data, the controller asks for the current value of the event number counter. The AUXbus controller will then check that all the slaves stay in sync with it.

AUXbus does not support either interrupt handling, or multi-master operations. Data transactions are unidirectional (Read only) and always performed in blocks. Hence, AUXbus is not a replacement for a general purpose protocol like the VMEbus. AUXbus, instead, is targeted to applications such as front-end readout in HEP experiment, where both event driven performances and real time response are crucial.

The protocol has been optimized to achieve high performances in block transfer mode. It overcomes the limitation of a fixed length packet and introduces a sparse data scan labeled with an event number. Used together with the VMEbus, it offers the needed power to squeeze a subevent builder in each front-end DAQ crate.

3 ROCK: the AUXbus controller

The ROCK is a VME double height slave board with A32/D08(O), D16, D32 and D32:BLT transfer capability. It is master on the AUXbus, where it supports the three phases of the protocol.

Data gathered on the AUXbus is arranged on a frame by frame basis into the Data and/or Event dual port FIFO banks. Each frame (Fig. 1) begins with a Start-Of-Frame word (SOF) containing the event number and the crate address. This is then proceeded by the data words, followed by an End-Of-Frame word (EOF) containing a parity to check data integrity at system level. Event numbers with no data are framed in empty structures, composed of only a SOF word, with a bit indicating an empty frame. The parity is built from the SOF word and all the data words. It is formed in the LLRC method, in which even parity is calculated in the bit direction, over a burst. Thus, for example, bit N of the LLRC is the XOR of bit N of all the words. The VME interface contains the control registers for the board housekeeping and is the gateway to the FIFO banks. Through the VMEbus the main features and modes of the boards can be programmed and monitored. Any monitoring can be done without interrupting or disturbing the AUXbus transactions.

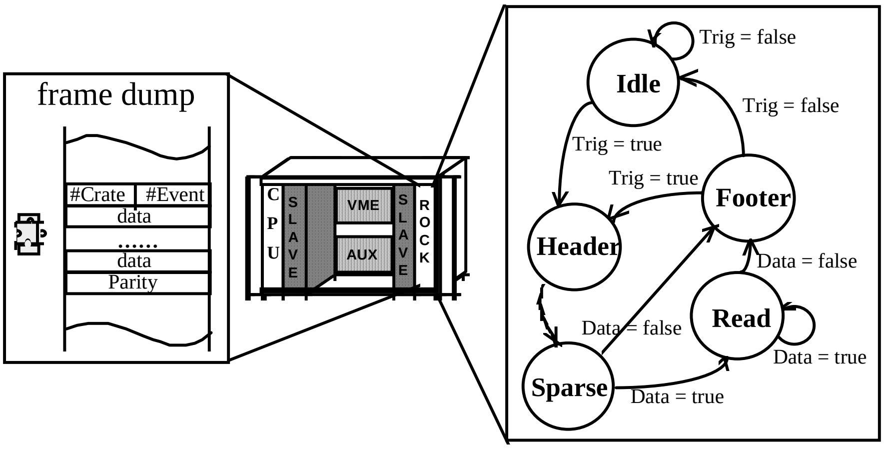

Fig. 1 Event driven flow and data framing on AUXbus

The FIFO flag status pattern and the AUXbus' main parameters are displayed, as well as the present and the last arrived event numbers. Time-out logic can be enabled with different watchdog scales to detect hang up conditions on the AUXbus. In case of an error, the VME can read the present state of the AUXbus interface, along with synchronization-check information. The VME can also redirect the AUXbus data frames to a specific bank (Data or Event) or to both, choosing the bank in charge to assert the empty/full conditions. In a typical application, AUXbus frames are pushed into the Data FIFO bank to be transmitted to the next DAQ level. However, frames can be sampled by the VME using the Event FIFO bank without slowing down the AUXbus traffic. This allows data "snooping" to monitor the system performances in real time. The ROCK is also able to measure on demand, the elapsed time for an event number processing. Both features make the system bandwidth control feasible without using extra hardware.

The ROCK performs data acquisition on the AUXbus, subevent assembly and data framing without any software overhead. All the operations are event driven and do not require CPU intervention. The VMEbus is used only to configure the board, to monitor and display the bandwidth usage and for diagnostic purposes. More details on the architecture of the ROCK can be found elsewhere [3,4,5].

4 The AUXbus backplane

Since the AUXbus is a high performance protocol, the transmission line effects introduced by the interconnections must be taken into account. In the same fashion as the VSB, AUXbus uses the 64 lines available on rows A and C of the VME/J2 connector and adopts TTL line drivers. The requirement for a clean electrical environment makes unfeasible a solution based on a flat cable interconnection between slots or a piggy-back module plugged on the rear side of a VME standard J2 backplane. In order to achieve the highest signal integrity, a monolithic 21-slot backplane has been designed, which merges the VMEbus (A32/D32), AUXbus and a modified version of the CERN's JAUX subsystem. The rightmost slot is reserved for the AUXbus controller and the sixteen AUXbus slave boards occupy the adjacent slots. The last slot of the AUXbus section has been foreseen for monitoring purposes and to send trigger signals onto the JAUX subsystem. The leftmost slot is reserved for the VME Master and the two adjacent slots are VMEbus-only. All the signal lines are constructed using microstrip technology, with an original trace impedance of about 110 Ohm. The additional capacitance introduced by the through holes and connector pins lowers the effective characteristic impedance of the lines to about 75 Ohm. AUXbus lines are terminated in the same fashion as the VMEbus. The AUXbus backplane has been fully characterized using the Time Domain Reflectometry (TDR) technique (Fig.2)

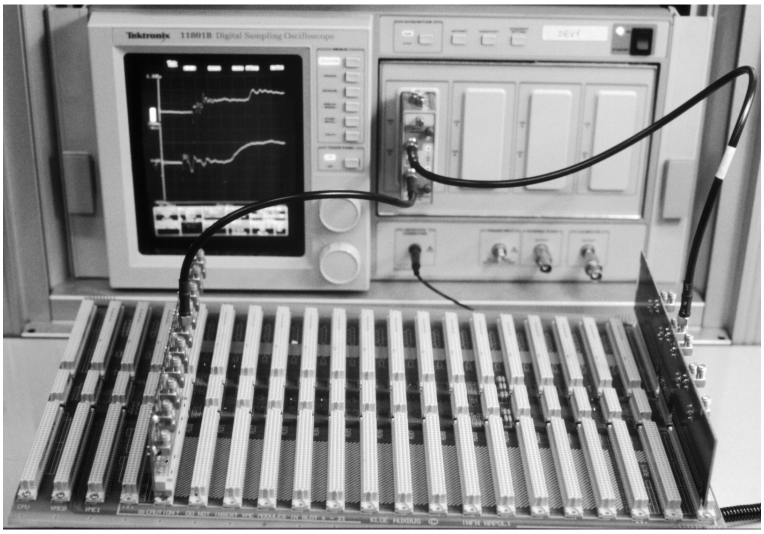

Fig.2 - The AUXbus backplane under TDR test

TDR is a method to calculate line impedance[6] measuring the incident and reflected wave caused by the impedance mismatch between the backplane line and the known impedance of the test environment. The TDR analysis detects impedance modulations due to design rule violations in the PCB manufacturing. It makes it also possible to evaluate the impact of different connectors on the effective characteristic impedance of the signal lines. In this way, the backplane design has been fine tuned during the prototyping stage to obtain the highest performances.

5 Conclusions

The AUXbus architecture permits true event driven operations by moving the most time consuming operations - such as the subevent assembly and the data framing - to hardware and to the protocol itself.
AUXbus has been adopted by the KLOE experiment at the DAΦNE Φ-Factory of the INFN National Laboratories, Italy. For this experiment ADC, TDC and general purpose digital I/O boards have been developed by companies and academical institutions, following the AUXbus specifications. In our tests, we have achieved sustained data transfer rates in excess of 15 MTransfer/s. Such performances do not depend on the CPU power, or on the software, but only on the slave's response time.

6 References

1. The KLOE Collaboration, KLOE a general purpose detector for DAΦNE LNF-92/019 (IR) 1992.
2. The KLOE Collaboration, The KLOE data acquisition system - addendum to the KLOE technical proposal LNF-95/014 (IR) 1995.
3. A. Aloisio et al., ROCK: The Readout Controller For The KLOE Experiment, IEEE Trans. on Nucl. Science, 43, 1, 167 (1996).
4. A. Aloisio et al., Custom Busses for large scale data acquisition systems, IEEE International Conference on Electronics, Circuits and Systems, ICECS 1996 Rodos (Greece) vol.2, p.1155.
5. A. Aloisio et al., Level 1 DAQ System for KLOE in Computing in High Energy Physics, 1995, Rio de Janeiro (Brazil), p.371.
6. IEEE Std 1194-1991, IEEE Standard for Backplane Electrical Performance.