ISTITUTO NAZIONALE DI FISICA NUCLEARE

Sezione di Napoli

INFN/TC-01/15
23 Ottobre 2001

THE ARGO MEMORY BOARD

A.Aloisio[1-2], S. Cavaliere[1-3], P. Di Meo[1], V. Masone[1], S. Mastroianni[1-3], L. Parascandolo[1], P.Parascandolo[1]

[1] *INFN Sezione di Napoli*
[2] *Università del Sannio, Benevento*
[3] *Università degli Studi Federico II, Napoli*

Abstract

This note describes the ARGO Memory Board module designed to be used within the Data Acquisition (DAQ) system of the ARGO detector at Yanbajng (Lhasa, Tibet).

Published by ***SIS–Pubblicazioni***
Laboratori Nazionali di Frascati

1. Introduction

The ARGO detector at Yanbajng (ARGO-YBJ experiment) is aimed to perform gamma ray astronomy and gamma ray burst physics in the range 100 GeV – 500 TeV. The detector [1 – 2] consists of a single layer of RPC (Resistive Plate Counters) covering an area of ~ 6500 m^2 and providing a detailed space time picture of the shower front. The detector is organised in modules of 12 chambers called clusters and consists of 130 clusters in the central part and 24 clusters in the guard ring. In each cluster the 960 pertaining strips are sampled in the local stations with a time resolution of 1 ns by custom digital multi-hit TDCs [3 - 4].

The trigger signal acts as a common stop for all the digital multi-hit TDCs reading the detector's strips in the Local Stations (LS). The ARGO-YBJ adopts an event driven data collection scheme whose key feature is to allow block oriented data transfers and read-out cycles labelled by trigger number. In each cluster the Local Station assembles a data frame containing an incrementing event number, the address of the fired detector's strips and all the timing information retrieved from the TDCs. All the Local Stations are connected with a star-like custom network to the Central Station where the main trigger logic as well as the read-out system are located.

Data frames originated at the cluster level are pushed into the ARGO Memory Board (AMB) entering the Level-I read-out system.

A two layer read-out architecture originally developed for the KLOE experiment has been adopted. The Level-I environment is based on crates equipped with VME bus and AUXBUS a custom high speed bus. In each crate a read-out controller – the ROCK [5-6] – manages the data transfer through the AUXBUS. In order to be scalable the system is divided in modular structures of VME crates tied by a vertical connection. These chains are made of up to 8 crates, each with up to 16 AMBs and the ROCK. The vertical connection links all the ROCK boards to the Level-II chain controller the ROCK Manager [7]. The ROCK performs crate level read-out and gathers data from the AMBs using the AUXBUS. In the same fashion the ROCK manager performs chain level read-out collecting the data frames belonging to a given event number from all the ROCKs.

2. The cluster data frame

The ARGO collaboration has chosen to acquire data in an event driven fashion. For each first level trigger issued the Local Stations electronics associate an 8 bit wide event number common to all the clusters which aligns data. Clusters with no data both from TDCs and strips transmit a special empty frame.

At any time it is possible to perform a check of the current event number the clusters are pointing to. A SYNC command can be issued to the same time to all the clusters. In response the Local Stations upon terminating their current trigger activity will send a special SYNC frame similar to the empty frame.

A typical block of data from the Local Stations consists of an header followed by the event number word (i.e. the 8 bit counter that is incremented at each trigger) plus TDCs words (TDC event number followed by data and addresses). TDC no. 0 is transmitted first. Then the fired RPC strips are transmitted. The frame is closed by three words: STOP, WORD_COUNT and a

word of vertical parity. In the following table an event consisting of 4 TDC having respectively 2 hits, 3 hits, 1 hit and 2 hits with 7 STRIP is shown:

```
START              0000  1010  1010  1010  - 0AAA\H
EV_NO. Created     0111  0000  XXXX  XXXX  - 70XX\H
EV_NO. TDC0        0100  0000  XXXX  XXXX  - 40XX\H
TDC_DATA           0010  0XXX  XXXX  XXXX  - 2XXX\H
TDC_ADR            0000  000X  XXXX  XXXX  - 00XX\H
TDC_DATA           0010  0000  XXXX  XXXX  - 20XX\H
TDC_ADR            0000  0000  XXXX  XXXX  - 00XX\H
EV_NO. TDC1        0100  1000  XXXX  XXXX  - 48XX\H
TDC_DATA           0010  1XXX  XXXX  XXXX  - 2XXX\H
TDC_ADR            0000  1XXX  XXXX  XXXX  - 0XXX\H
TDC_DATA           0010  1XXX  XXXX  XXXX  - 2XXX\H
TDC_ADR            0000  1XXX  XXXX  XXXX  - 0XXX\H
TDC_DATA           0010  1XXX  XXXX  XXXX  - 2XXX\H
TDC_ADR            0000  1XXX  XXXX  XXXX  - 0XXX\H
EV_NO. TDC2        0101  0000  XXXX  XXXX  - 50XX\H
TDC_DATA           0011  0XXX  XXXX  XXXX  - 30XX\H
TDC_ADR            0001  000X  XXXX  XXXX  - 10XX\H
EV_NO. TDC3        0101  1000  XXXX  XXXX  - 58XX\H
TDC_DATA           0011  1XXX  XXXX  XXXX  - 3XXX\H
TDC_ADR            0001  100X  XXXX  XXXX  - 1XXX\H
TDC_DATA           0011  1XXX  XXXX  XXXX  - 3XXX\H
TDC_ADR            0001  100X  XXXX  XXXX  - 1XXX\H
STRIP              1XXX  XXXX  XXXX  XXXX  - XXXX\H
STRIP              1XXX  XXXX  XXXX  XXXX  - XXXX\H
STRIP              1XXX  XXXX  XXXX  XXXX  - XXXX\H
STRIP              1XXX  XXXX  XXXX  XXXX  - XXXX\H
STRIP              1XXX  XXXX  XXXX  XXXX  - XXXX\H
STRIP              1XXX  XXXX  XXXX  XXXX  - XXXX\H
STOP               0000  0101  0101  0101  - 0555\H
WORD_COUNT         0000  0000  0111  1111  - 001F\H
PARITY
```

An empty event has the following format:

```
START              0000  1010  1010  1010  - 0AAA\H
EV_NO. Created     0111  0000  XXXX  XXXX  - 70XX\H
STOP               0000  0101  0101  0101  - 0555\H
WORD_COUNT         0000  0000  0111  1111  - 001F\H
PARITY
```

The SYNC check format is the following:

```
START              0000  1010  1010  1011  - 0AAB\H
EV_NO. Created     0111  0000  XXXX  XXXX  - 70XX\H
STOP               0000  0101  0101  0101  - 0555\H
WORD_COUNT         0000  0000  0111  1111  - 001F\H
PARITY
```

3. Data transmission

Due to the rather long distance between the front-end station and the DAQ station (about 75 meters) a single ended transmission is not feasible. A balanced transmission of the data employing a differential receiver has the clear advantage of a high common mode voltage rejection, reduced EMI and improved speed capabilities. Therefore we have opted to transmit data differentially with the PECL format at 10 MHz. Quad differential drivers and receivers from Lucent are used [8]. The receivers have an internal protection circuit which significantly improves the ESD characteristics of the component. The line terminating resistors have a 120 Ω value.

A 40 line twisted ribbon cable links the Local Station to the AMB. A sixteen bit data word, clock, two control lines, and two GND are transmitted in parallel. When a first level trigger is issued each Local Station transmits blocks of data of variable length.

Via the AMB two control signal, Test_Sync and Reset, are forwarded to the Local Stations. The Test_Sync signal is width sensitive. Whenever its width is 200 nsec the front-end cluster will produce the SYNC frame. Should its width be much larger (≅20 μsec) then the front-end cluster will issue a 20 words fix test pattern.

At the end of the frame each Local Station transmits further four clocks with the data lines at zero. These clocks are requested for properly reset the internal registers of the AMB.

4. AMB Overall Block Diagram

The AMB is a VME double height slave board with A32/D16, D32 data transfer capabilities. This board adopts a regular and flexible resources allocation scheme and the physical layout is shown in Fig. 1.

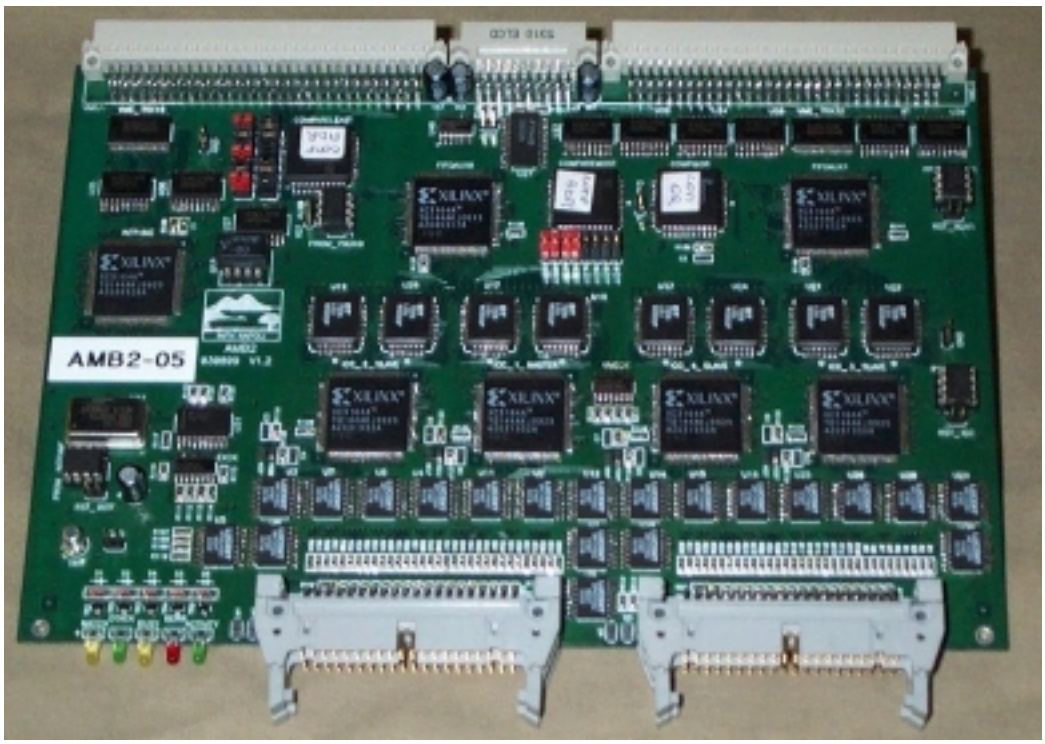

Fig. 1 AMB module.

The block diagram of Fig. 2 shows four identical input slice each handling a 16 bit data path. The line receivers output is fed to four Front End FE-FPGA (3164A Xilinx) each serving a cluster input which redirect the data flow toward an asynchronous dual port 8K word FIFO bank.

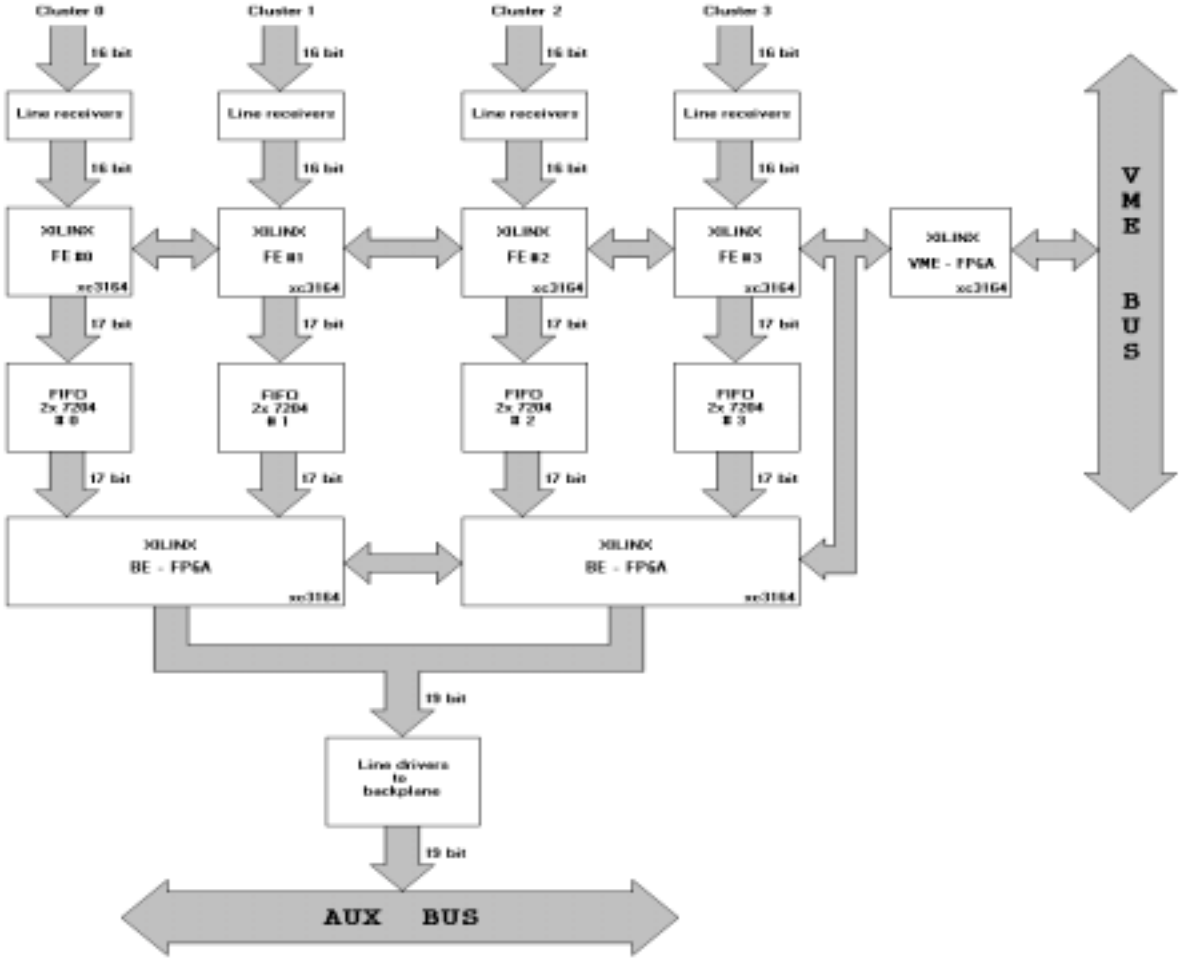

Fig. 2 Block Diagram

The FIFO bank de-couples the slices'data traffic from the core inner logic of the board, the two Back End BE-FPGA. The BE-FPGA receives the FIFO flags from all the slices and manages the read-out accordingly gaining full access to all the AMB input channels. Processed data can be directed on the AUXBUS to be read-out by the local Level–I controller or on the VME bus. The VME-FPGA contains most of the internal registers to program the board and acts as a gateway to access the internal resources.

5. The Front End (FE) FPFA

Each Front End FPGA receives the data Cluster and parses the data stream syntax (Fig. 3).

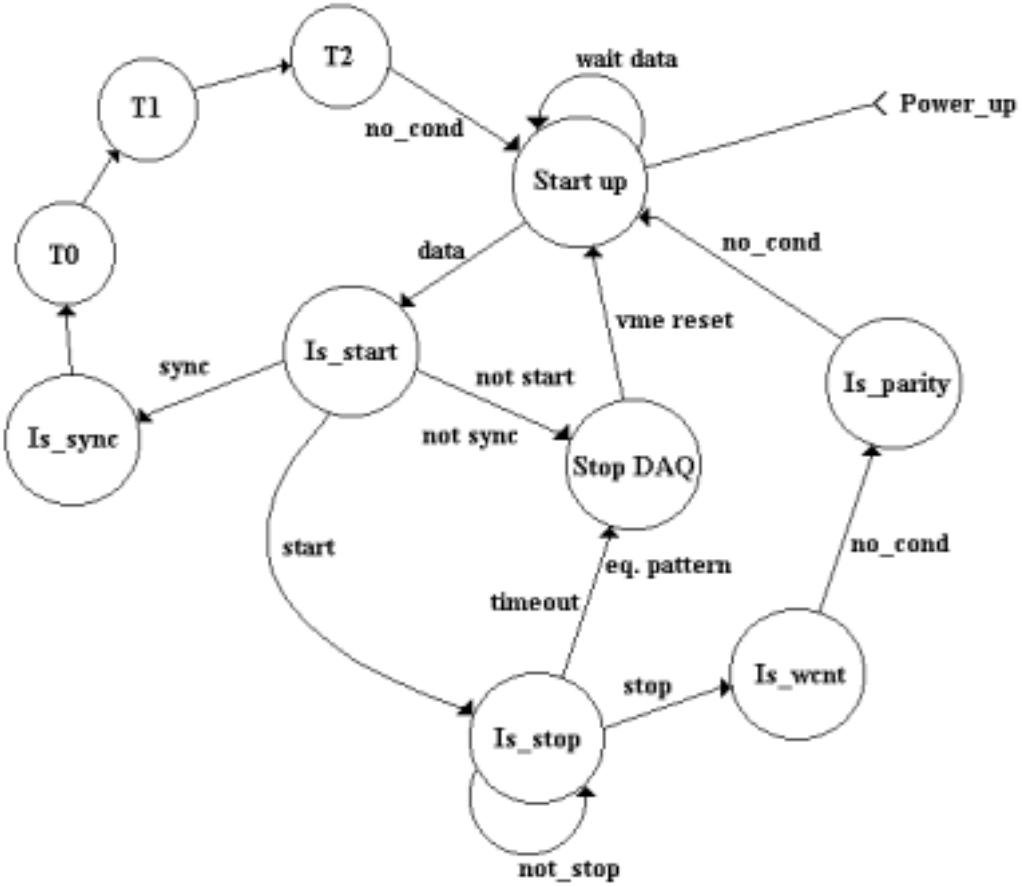

Fig. 3 FE-FPGA state machine

The first data word triggers the state machine to check the header pattern. At the same time a Busy flag is set to notice activity. The Busy flag is then reset after the parity word is detected. In case of a valid header the state machine enters in the *is_stop* state where the input data is checked for a match with the stop pattern. Actually the stop pattern must be received within 1024 words after the start otherwise an error flag is set. The error flag is also set whenever the same input pattern is detected on three consecutive words.

Upon an acknowledgement of a valid stop pattern the state machine counts the length of the data frame and performs a vertical parity check. In case of failure a check bit set to one is added to the pattern to be written into the FIFO.

The empty frame and the sync frame are processed in a similar fashion.

6. The Back End FPGA

These two FPGAs strip off empty frames and collect the four fragments for a given event number building up a 4 Cluster image of the detector′s response. The two FPGAs work in parallel, so that the most time consuming operations – such as sub event assembly and data framing – are optimised. The BE single channel functionality is depicted in Fig. 4.

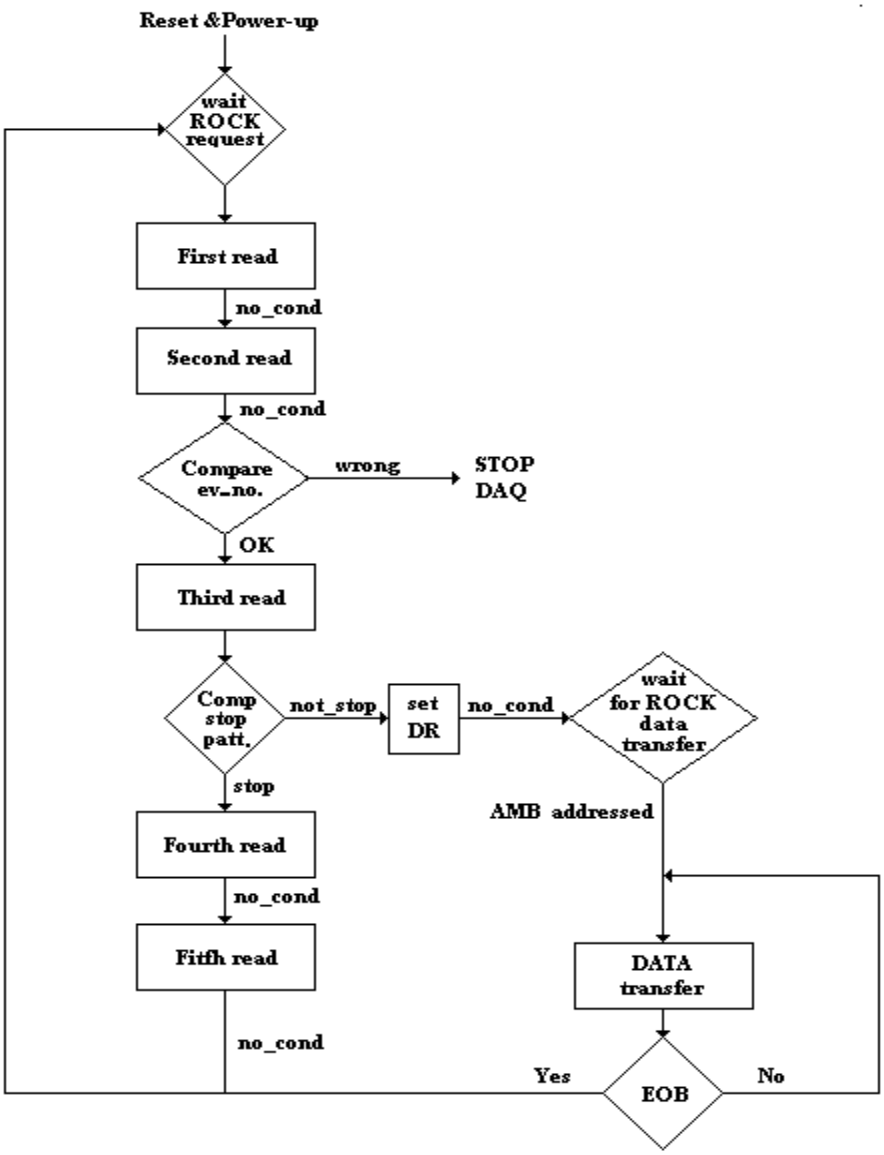

Fig. 4 BE Flow chart

Data are read-out from the FIFO at the ROCK request. Data acquisition from the ROCK proceeds in two steps: the trigger broadcast cycle where an event trigger number is issued and the data transfer cycles where the AMB is requested to output the data.

During the trigger broadcast cycle, the two BE-FPGAs read from the FIFO the frame header and the event number. The third word distinguishes between a data frame and an empty frame. In case of a data frame a Data Ready (DR) flag is sent to the ROCK, signifying that at least one cluster has data pointing to the event number in progress. In the case of an empty frame, the third word contains the stop pattern followed by a word count and parity. If the four clusters handled by an AMB board all transmit empty frames for a given event number the Data Ready flag to the ROCK is not asserted.

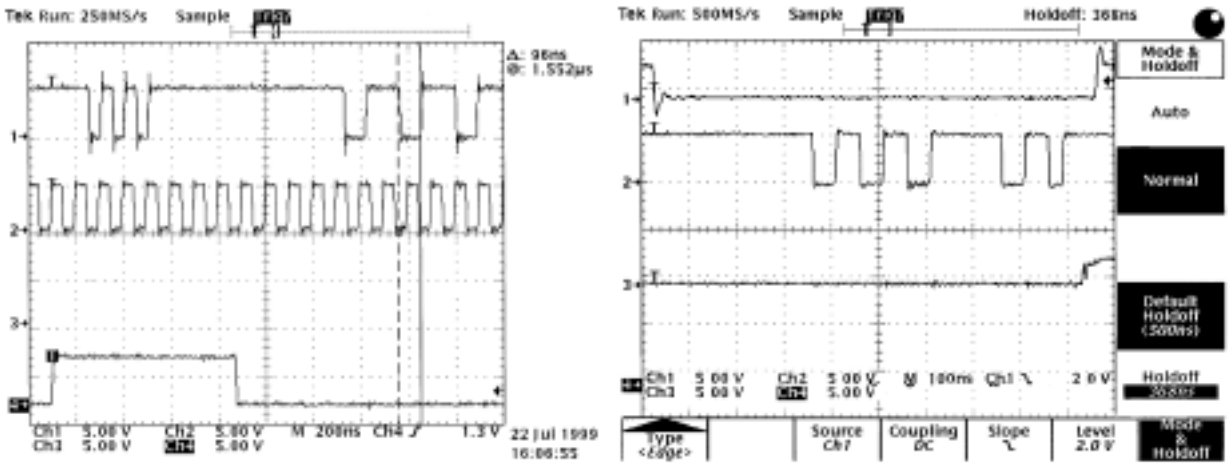

Fig. 5 Trigger broadcast and empty frame suppression.

The data transfer cycle starts when the ROCK places a valid geographical address onto the AUXBUS and drives low the address strobe line AS. Each AMB will decode the address. The addressed module starts sending data using a handshake VME like protocol. The Rock requests data on the high to low transition of XDS. The AMB outputs the data and then drives XDK low. The ROCK recognises that the data is valid and latches with the raising edge of XDS. The AMB recognises that the ROCK has latched the data and raises XDK. The process continues until the ROCK receives an End of Block flag (EOB).

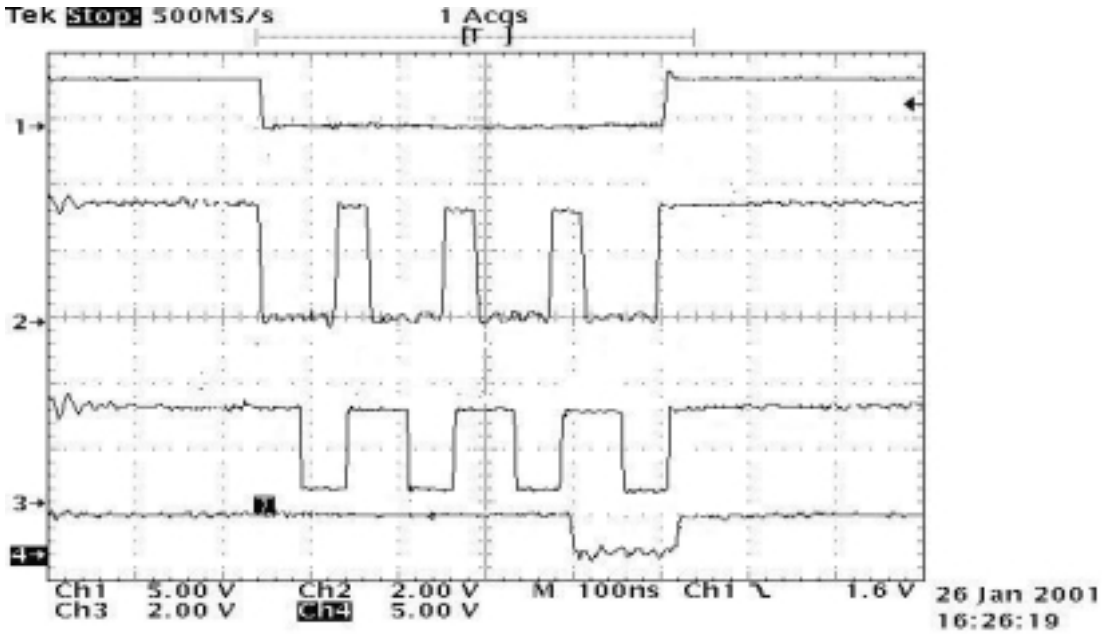

Fig. 6: AS, XDS ---XDK handshake and EOB flag.

To assure an orderly transfer of the cluster's data to the ROCK a token in token out mechanism has been introduced. Data from cluster zero is transferred first followed by clusters 1, 2 and 3. If no

data is available from a given cluster the pertaining empty frame is suppressed. The EOB flag is set by the last cluster sending data.

7. The SYNC cycle

A special header's code is assigned to the SYNC header. When the header field contains the value (0AAB\H) the event numbers sent by the clusters are saved into each of the FE-FPGAs. No data is written onto the FIFO.

When the ROCK begins a SYNC cycle on the AUXBUS each AMB verifies that the four event number match. If this check is successfully the event number is transferred to the ROCK completing the cycle, otherwise an error flag is set to request attention.

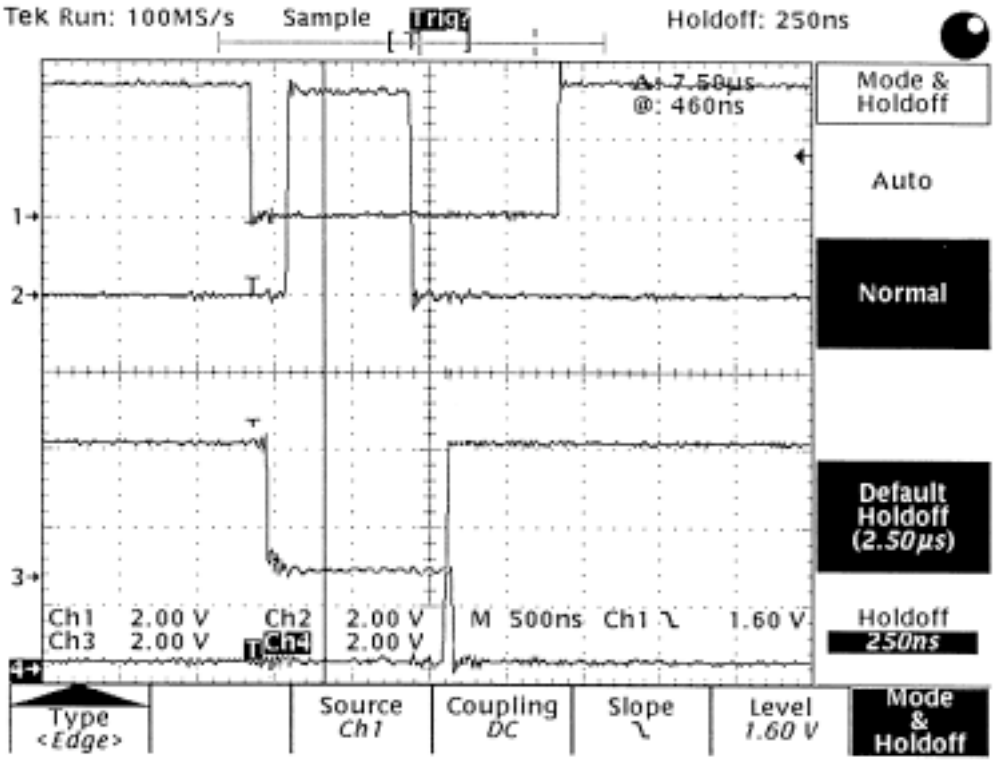

Fig. 7 SYNC cycle

8. The AMB status flags

Two flags are produced by the AMB and forwarded to the ROCK. XBUSY is the logical OR between the four Busy signals produced by the FE-FPGAs and the four Half-Full flags of the FIFO, XBERR flags a corruption either in the data frame syntax parsed by the FE-FPGAs or a mismatch between the event number requested by the ROCK and the one received by the clusters.

The presence of XBUSY inhibits further trigger production. XBERR instead, halts the entire data flow in order to detect the reason of the failure.

9. The VME interface

Data from AMB can be transferred across the AUXBUS as in normal operations or using the VME for stand alone operations. The AMB also supports diagnostic mode which allows the user to write test patterns in each input slices, emulating the Local Station output. In this way an extensive

test of the entire data flow can be performed, including the frame parsing in the FE-FPGA, the FIFO integrity and the event building in the BE FPGA.

Conclusions

The AMB has worked successfully in the lab and actually has been already installed on the experiment for data taking.

References

1. "The ARGO full coverage detector" Proc. of the 25th IRC Durban (1977), vol.5, pg. 265
2. "Physics with the ARGO detector" Proc. of the 25th IRC Durban (1977), vol.5, pg. 269
3. M. Passaseo, E. Petrolo, S. Veneziano "A TDC integrated circuit for drift chamber readout" NIM A367 (1995) 418-421.
4. M. Passaseo, E. Petrolo, S. Veneziano "Design of a multichannel TDC integrated circuit for drift chamber readout" Proc. of the International Conference on Electronic for Particle Physics – LeCroy Research System, Chestnut Ridge, May 1995.
5. A.Aloisio, S.Cavaliere, F.Cevenini, D.Fiore, M.Della Volpe, L. Merola, P. Parascandolo "Rock: the Readout Controller for the KLOE Experiment" IEEE transaction on Nuclear Science, Vol. 43, No.1, February 1996, pp.167-169.
6. A.Aloisio, F.Cevenini, D.della Volpe, L. Merola, P. Parascandolo "Custom solution for a data readout architecture" CHEP94 - San Francisco (USA) - Proc. 88 (94).
7. A.Aloisio, S.Cavaliere, F.Cevenini, D.Fiore, M.Della Volpe "Level 1 DAQ for the KLOE Experiment" Proceedings of the CHEP 95 - Conference on Computing in High Energy Physics - pp.371-375. Rio De Janeiro, 1995.
8. Lucent Technologies, Data Transmission Devices, AP99-005HSI, 1998.