FRITZ: A Universal Fastbus–VME Interface

Rainer Wanke [a], Klaus Honscheid [a], Charlie Strohman [b]

[a] *Ohio State University, Columbus, OH 43210*

[b] *Cornell University, Ithaca, NY 14853*

A Fastbus-VME interface, called FRITZ, has been developed for the data acquisition system of the CLEO III detector. As a Fastbus board with an on-board two-slot VME backplane it allows Fastbus access using a regular VME CPU module. Single Fastbus operations are possible as well as a fast crate read-out via Direct Memory Access.

Key words: VME; Fastbus; DAQ; FRITZ; CLEO

The data acquisition system of the CLEO III detector [1] will be based on the VMEbus. VME CPU modules will read out the front-end crates and transfer the data to event builders and level 3 triggers. The expected level 1 trigger rate is about 1 kHz, leaving 1 ms for the data acquisition system to process a single event. While VME data boards will be used for the front-end electronics of some sub-detectors, the crystal calorimeter, the muon system, and the drift chambers will be read out by LeCroy TDC 1877 Fastbus modules, which provide the necessary performance [1]. In order to maintain the advantages of a VMEbus based data acquisition system, such as readily available processor modules, we have developed FRITZ. The FRITZ module (F*astbus* R*eadout* I*nterface with data* T*ranslation and* Z*ero-suppression*) provides an interface between Fastbus and VMEbus that makes Fastbus look like VME. Fastbus operations are performed with a regular VME CPU module. The advantages of this approach over conventional solutions are obvious:

- Only one type of CPU module is needed both for VME and Fastbus crates.
- We have to buy only one real-time operating system and one development environment (compiler, debugger, etc.).
- Only one type of data link to the event builder is necessary.
- The read-out and monitoring software is more or less identical.
- Upgrades are easy: As soon as higher performance VME CPU modules become available they can be used for Fastbus as well.

FRITZ consists of a standard size Fastbus printed circuit board containing not only the Fastbus interface logic and ECL drivers but also a complete VME interface including a two-slot VME backplane to plug in a 6U VME CPU module (Fig. 1). A general picture of the data flow inside the FRITZ board is shown in Fig. 2. The logic of FRITZ is divided into several distinct modules, such as VME Master and Slave Interfaces, a Fastbus Master Interface, sequencer logic, etc. Each of these modules is typically implemented in a single Altera gate array (FPGA). The internal modules are connected with each other via a local bus, which is separated from both VME and Fastbus by latches controlled by the VME Slave and the Fastbus Master Interface, respectively. The VME Slave Interface controls the VME access to FRITZ. It determines if FRITZ is addressed, selects the appropriate internal module, performs an internal data cycle, and finally acknowledges the VME CPU.

A VME CPU module inserted into FRITZ has two choices to access Fastbus modules. These are the operation as a basic Fastbus Controller using programmed input/output and, secondly, the execution of a fast crate readout via Direct Memory Access (DMA). In the DMA mode Fastbus and VME

[1] In case of charge measurements $Q \to T$ chips convert the output to time intervals which are fed into the TDC's.

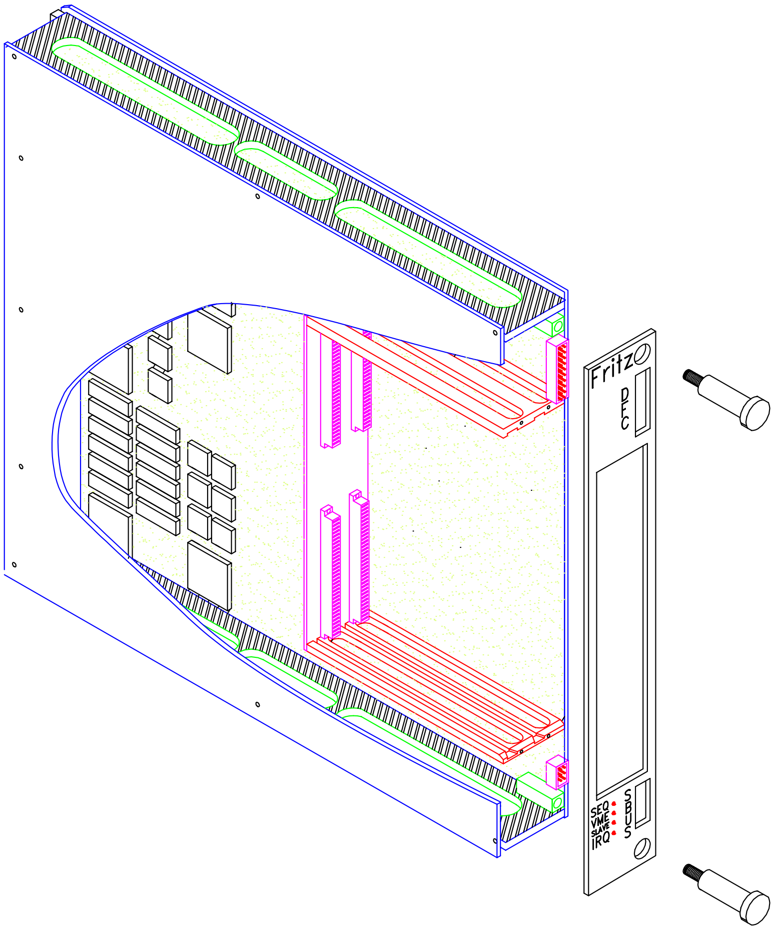

Fig. 1. **General layout of FRITZ.**

block transfers are used. All data written to and read from the Fastbus backplane are passed through ECL/TTL level shifters placed close to the Fastbus backplane.

In programmed I/O mode, every action of FRITZ is performed by single VME read/write cycles to the FRITZ internal VME slave interface, which then does read/write cycles on the internal bus to the Fastbus master interface. This means each single Fastbus action is executed through a complete VMEbus cycle. Apart from block transfers, all possible Fastbus operations can be performed in programmed I/O mode. The type of Fastbus action to be done is defined by the first few VME address lines. The accompanying VME data lines pass the Fastbus data including Fastbus addresses and status codes. All Fastbus control lines are mapped to a control and status register inside the Fastbus master interface. In this way, one can read the Fastbus return status codes, and also drive all Fastbus control lines separately, which is useful for testing and debugging.

To optimize performance during data taking we have implemented a DMA mode which allows the fast readout of a whole Fastbus crate. This mode includes "on the fly" sparsification and data conversion to reduce the amount of data written into the VME CPU memory. The heart of the DMA functionality is a list–driven sequencer logic using Fastbus block transfer cycles. Once the sequencer has been triggered by an appropriate VME cycle or by an external read-out signal, it executes Fastbus operations according to commands stored in an on-board list memory. Except block transfer write cycles, which are not needed for our purposes, all Fastbus operations can be performed. In DMA mode the internal bus lines are not used at all, instead Fastbus data are directly piped into an on-board FIFO, which decouples the Fastbus side from VME operations. Each single sequencer cycle (from one Fastbus DK transition to the next DS transition) is performed in less than 40 ns. Taking 85 ns as a typical response time for the TDC 1877 modules, we can achieve a read-out time of 125 ns per data word corresponding

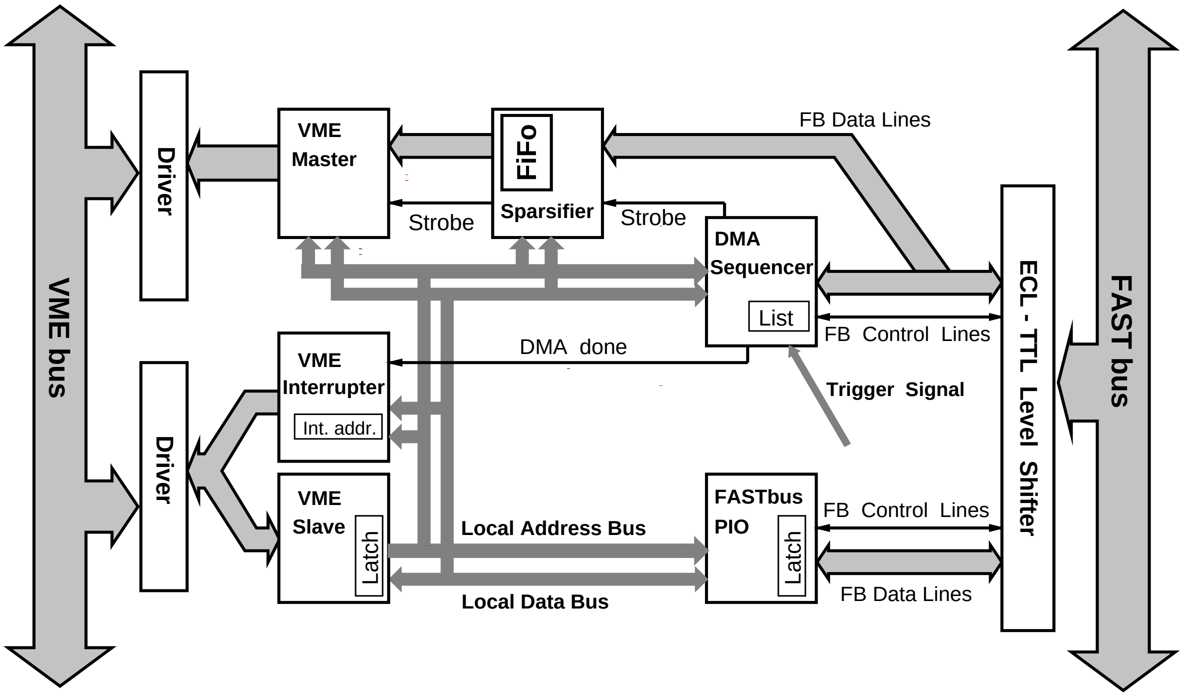

Fig. 2. **Block diagram of data-flow in FRITZ.**

to a data transfer rate of about 30 MByte/s. During transfer into the FIFO, the TDC data are sent through a sparsification module, which can perform the following operations on the data:

– TDC data below a channel-dependent pedestal can be thrown away.
– TDC channel addresses can be converted into more physical calorimeter and drift cell identifiers.
– For TDC data lying within a certain channel-dependent range, a quality bit can be added to the data to simplify the later off-line analysis.
– Header words of the TDC data can be rejected.

Several look-up tables are used to store the pedestals, new identifiers, etc. The sparsification reduces the event size by a factor ≥ 10 during normal data-taking. As soon as the first data word arrives in the FIFO, the VME Master Interface retrieves the data and requests VMEbus mastership. When mastership has been granted, it writes the data into the CPU memory using VME block transfer mode. The VME Master Interface acts completely independent from the VME Slave Interface. If all DMA actions are terminated, the sequencer notifies the on-board VME Interrupter module, which then issues a VMEbus interrupt and answers to the following VME interrupt cycle.

We thank our colleagues in the CLEO III Data Acquisition and Trigger group for their support. This work is sponsored by the U.S. Department of Energy, the Alexander von Humboldt Foundation and the Alfred P. Sloan Foundation.

References

[1] Overview talk by Andreas Wolf at this conference.