Complete Parallel Readout VME DAQ System

H. Baba[1], T. Ichihara[1], T. Ohnishi[1], K. Yoshida[1], Y. Watanabe[1], S. Ota[2], S. Shimoura[2], R. Yokoyama[2], S. Takeuchi[3], D. Nishimura[4] and A. Tokiyasu[5]
[1]RIKEN Nishina Center, Wako, Saitama 351-0198, Japan
[2]Center for Nuclear Study, University of Tokyo, Wako, Saitama 351-0198, Japan
[3]Department of Physics, Tokyo Institute of Technology, O-okayama, Tokyo 152-8551, Japan
[4]Department of Physics, Tokyo University of Science, Noda, Chiba 278-8510, Japan
[5]Research Center for Electron Photon Science, Tohoku University, Sendai, Miyagi 982-0826, Japan

In RIKEN RIBF, a new low-cost VME controller has been developed. The noteworthy feature is that data readout can be parallelized even VME modules are installed in the same VME crate. Therefore, the performance of the VME-based DAQ system will be maximized. In this contribution, we report the specification of the developed VME controller and the performance of the complete parallel readout VME DAQ system.

RIKEN RIBF is a nuclear physics research facility generating unstable nuclei. The length of the beam line is over 100 meters, and several detector sets are installed at distant places. Several CAMAC and VME front-end systems are installed along the beam line. Individually taken data are merged by RIBF DAQ system [1]. The dead time of the system is determined by the number of readout channels per CAMAC/VME controller. The bus frequency is limited by the system specification. To achieve a higher speed, the number of readout channels and the interrupt latency should be reduced. Recently, FPGA-based bus controllers have been distributed by companies. CC/NET [2] and VM-USB [3] are FPGA-based CAMAC and VME controllers that can run as a list sequencer. They can achieve the intrinsic readout speed of buses. Moreover, the interrupt latency is quite shorter than that of the usual PC-to-VME system. These controllers are enough good. However, it is almost impossible to install many controllers for reducing the readout channel per controller, due to the cost. In this study, we have developed a new truly low-cost VME controller to maximize the performance of the VME-based DAQ system. This controller is named as "MOCO" that stands for mountable controller for VME.

A photograph of MOCO is shown in Fig. 1. This controller mainly consists of FPGA (Xilinx, XC3S50AN, Spartan 3AN) and USB2.0 slave (FTDI, FT2232H Hi-speed Dual USB UART/FIFO) chips. The implemented VME functions are listed below.

- Address mode : A32, A24, A16
- Program I/O : D16 Read/Write, D32 Read/Write
- Block I/O : D32 Block-Read
- Interrupt detection (IRQ)
- Buss error detection (BERR)

In addition to these VME functions, MOCO has auxiliary 4-ch LVDS input/output (Aux I/O) ports. These Aux I/O ports are used to synchronize with other systems by receiving a trigger and generating a busy (or end-of-busy) signals. MOCO can be controlled by a usual PC through the USB port. When MOCO receives a readout command from the PC, VME data is stored into the FIFO memory of the USB chip. MOCO has some special operation modes to continuously read data from the module without PC intervention. An example of the continuous-readout operation mode is as follows:

1) MOCO waits the trigger signal from Aux I/O
2) If the trigger is arrived, MOCO waits the IRQ signal from the module
3) When the IRQ signal is issued, MOCO starts readout
4) MOCO takes data until the preset number of readout cycle is reached or BERR is generated by the module
5) When the readout cycle is terminated, MOCO generates the end-of-busy signal through Aux I/O
6) Go back to 1)

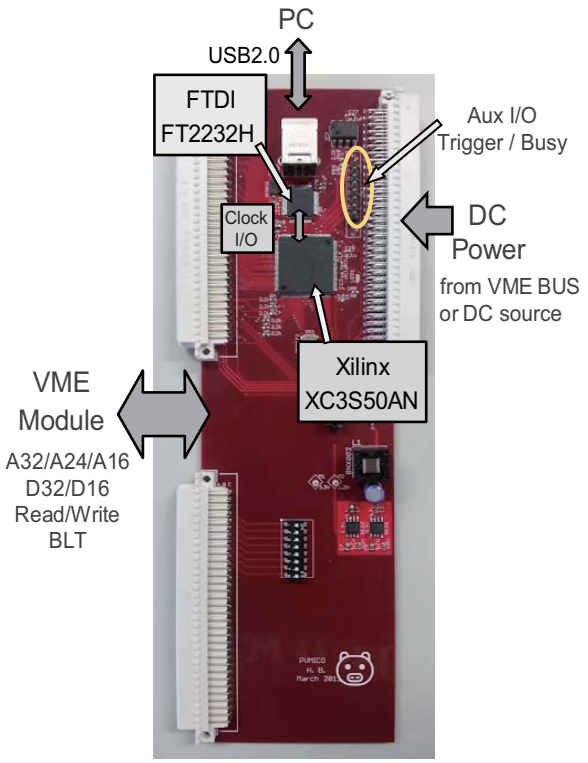

Fig. 1. Photograph of the mountable controller for VME (MOCO).

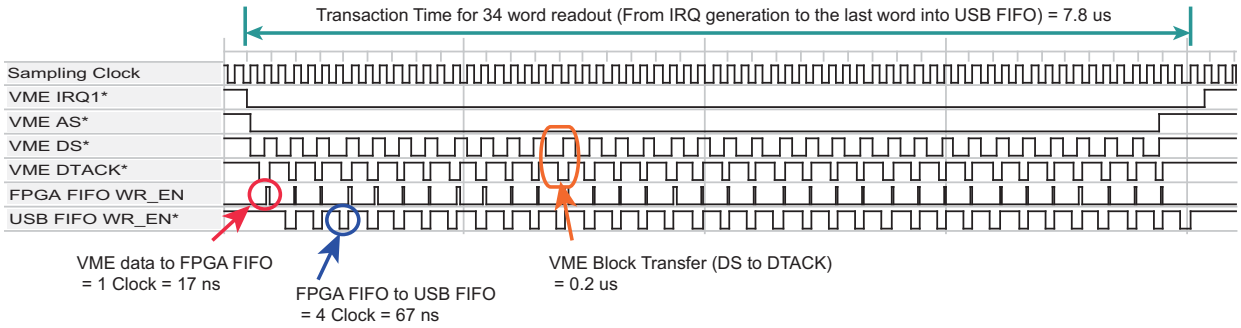

Fig. 2. Timing chart of the response of the interrupt signal and 34 data block readout.

The transaction time of the VME operation could be minimized with this special operation mode. It is also possible to implement more complex readout sequence in MOCO.

The transaction time of MOCO has been investigated. Figure 2 shows the timing chart of the interrupt response and the block transfer. The FPGA chip in MOCO is configured such that when the interrupt signal is generated, 34 data (34 $\times$ 32 bit = 272 bytes) are readout from a CAEN V792 QDC and transferred to the USB chip. The timing chart is obtained by ChipScope Pro [4], which is a debug tool to inspect the interior of FPGA. *VME AS** (Address Strobe), *DS** (Data Strobe), and *DTACK** (Data Acknowledge) are VME bus signals used to perform handshake. Note that an asterisk (*) following the signal name represents inverted signals. First, VME data are stored in the FPGA FIFO memory (*FPGA FIFO WR_EN*). Subsequently, data is transferred to the USB FIFO memory (*USB FIFO WR_EN**). The sizes of both FIFO memories are 4 kBytes. The transaction time of this sequence is only 7.8 μs (1 data readout $\simeq$ 0.2 μs). This speed is almost the maximum performance of CAEN V792 QDC. The worst interrupt latency is only 16.6 ns since the FPGA watches the *VME IRQ** signal every 16.6 ns (synchronized with a 60 MHz clock). This interrupt latency corresponds to a speed that is 1000 times higher than the speed of the usual PC-to-VME system.

MOCO can be inserted between the VME module and the VME bus. Figure 3 shows a photograph of the installation. This case, VME bus signals on the VME crate are ignored, and only power lines are used. Therefore, even multiple VME modules together with MOCO are installed in the same VME crate, it is possible to readout data in parallel.

Data from VME modules are accumulated by the PC through the USB 2.0 interface. The data transfer rate from MOCO to the computer server via Ethernet by Raspberry Pi 2 has been measured. The maximum rate was 64 Mbps. This rate is not as high as the VME bus specification. However, in case of usual ADC/QDC/TDC VME modules, this performance is enough high.

The production cost of the controller is about 18,000 JPY ($\simeq$ 150 USD). The price of the Raspberry Pi 2 is about 4,800 JPY ($\simeq$ 40 USD). In the total, the cost is 22,800 JPY ($\simeq$ 190 USD) to take data from 1 VME module. If the required data rate is not so high, multiple VME modules can be handled from the same PC. In this case, the system will be further low cost.

Fig. 3. Photograph of the installation in the VME crate. MOCO is inserted between the VME module and the VME bus.

This MOCO based "Complete Parallel Readout VME DAQ System" has been installed to acquire data from the beam line detectors in RIBF113 and RIBF79R1 experiments at RIBF RIKEN. These experiments were scheduled April 9–22th 2016 (RIBF113) and May 15–25th 2016 (RIBF79R1), respectively. In these experiments, 7 $\times$ CAEN V1190 TDC, 2 $\times$ CAEN V1290 and 1 $\times$ Niki-glass LUPO modules were installed together with MOCO. Typical total data rate from these 10 modules was 2.7 MBytes/sec. And the live time ratio was $>$ 99% with respect to generated physics triggers. This performance is more than 3 $\times$ better than the standard VME-based RIBF DAQ [1].

In summary, we succeed to develop the very low-cost VME controller (MOCO). Even the low-cost PC such as Raspberry Pi, it can transfer data at a sufficient rate. By installing multiple MOCO, the complete parallel readout VME DAQ system was successfully constructed.

References

[1] H. Baba et al., Nucl. Instrum. and Meth. A, vol. 616, pp. 65–68 2010
H. Baba et al., Proc. IEEE Nucl. Sci. Symposium Conf. Record 2008, 1384–1386
[2] Y. Yasu et al.: Comp. High Ene. Nucl. Phys. (2003) 24.
[3] *Wiener VM-USB User Manual.*
[4] *Xilinx ChipScope Pro Software and Cores User Guide.*