THE APPLICATION OF POWERPC/VXWORKS TO THE READ-OUT SUBSYSTEM OF THE BESIII DAQ

Tao Ning, USTC, Hefei, CHINA
Chu YuanPing, IHEP, Beijing, CHINA
Zhao JingWei, IHEP, Beijing, CHINA

Abstract

This article describes the prototype of the read-out subsystem which will be subject to the BESIII data acquisition system. According to the purpose of the BESIII, the event rate will be about 4000Hz and the data rate up to 50Mbytes/sec after Level 1 trigger. The read-out subsystem consists of some read-out crates and read-out computer whose principle function is to initialize the hardware, to collect event data from the front-end electronics after Level 1 trigger, to transfer data fragments from each VME read-out crate to online computer farm through two levels of computer pre-processing and high-speed network transmission. In the test model, the crate level read-out implementation is based on commercial single board computer MVME5100 running VxWorks operating system.

The article outlines the structure of the crate level testing platform which included hardware components and software components. It puts emphasis on the framework of the read-out test model, data process flow and test method in crate level. Especially, it enumerates key technologies in the process of design and analyses of the test result. In addition, results which summaries the performance of the single board computer from the data transferring aspects will be presented.

OVERVIEW OF THE DATA ACQUISITION SYSTEM FOR BES III

After running successfully for more than ten years, the BEPC e^+e^- collider will be upgraded for higher luminosity, which will be increased to $10^{33}cm^{-2}sec^{-1}$. Therefore, its detector BESII will be upgraded accordingly to take advantage of this increased luminosity, named BESIII.

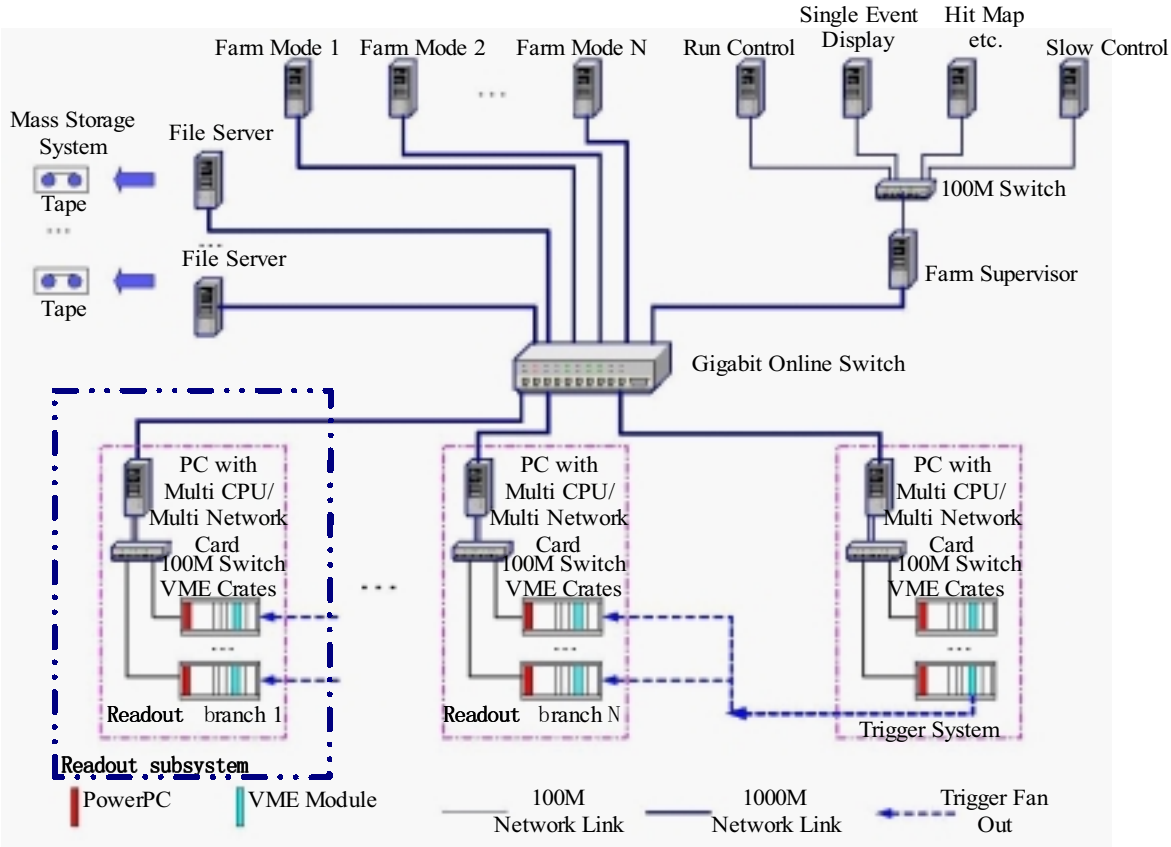

Figure 1: the framework of the BESIII DAQ system

Trigger rate and event size define the performance requirements for the data acquisition (DAQ) system. For peak luminosity, we expect a L1 trigger accept rate of up to 4000Hz. The total electronic channels are more than 40k. Assuming an expected more than 15% occupancy for MDC, about 17% for EMC, 1% for MUC and 10% for others, the estimated total data rate for the front-end readout will be up to 80Mbytes/s. The capabilities of the online farm processing and the tape-recording will be 50Mbytes/s and 40Mbytes/s. Respectively compared with BESII, the BESIII DAQ system gets a much higher design target both in scale and in performance, which requires a high-performance read-out subsystem.

The BESIII DAQ systems, responsible for the transportation of event data from the detector front-end electronic (FEE) to mass storage with a minimum of dead time are shown schematically in Fig.1. The interface between the DAQ system and FEE system, as well as the interface between the DAQ system and the trigger system are all defined.

In order to read out large amount of data from the FEE system as quickly as possible, the BESIII DAQ system adopts multi-level buffering, parallel processing, and high-speed VME readout and network transmissions techniques. The lowest level is the readout subsystem, containing one or more readout computers and some VME crates. Each crate consists of a standard set of components, including a commercial embedded processor (Motorola MVME5100 single-board computer) as a system controller and some other FEE readout modules (ADC and TDC). There are no more than 16 FEE readout modules in each crate. As soon as the data on the readout modules are ready, the single-board computer (SBC) is started by responding the interrupt from readout control module, autonomously reads the data from the FEE of the various detector systems and writes those through VME bus into the SBC memory, then pre-processes and assembles them into the accepted data fragment. Several readout crates aggregate the data and forward them to a readout PC via a switched 100Mbits/s Ethernet network, thus constituting a readout branch. All the readout branches, the online farm, and all the application and console server are connected via a gigabit switch that used for event building, filtering and then recording into a persistent media.

THE PURPOSE OF READOUT MODEL

To meet the requirement of the BESIII DAQ system and make sure that the SBC can read out data from the FEE within a critical deadline, we need such a readout system where a timely response by the SBC to external stimuli is vital. It was defined an embedded soft real-time system, where nothing catastrophic happens if some deadline are missed, but where the performance will be degraded below what is generally considered acceptable. The readout subsystem is finally used under a critical environment, and must therefore be carefully designed and validated before being into operation. Therefore, a readout test model was developed for prototyping the BESIII DAQ readout branch and for checking design correctness and characterizing performance and reliability. To keep stability and reliability of the readout system, all tasks that per average 4000 event data being read out and assembled and delivered must be done within 0.8 second and the processing time of CPU is limited within 0.6 second. That is to say, 20% redundancy and less than 60% CPU occupancy should be ensured. Furthermore, to make sure the high-speed readout, data are transferred from all the FEE readout modules to the SBC memory, that using a direct memory access (DMA) engine. At meantime, high-speed computing and memory-accessing and network transferring must be ensured.

In conclusion, the following function must be included in the readout test model: to collect event fragments from VME crates within limit time and assemble them into the tagged sub-event packages; to control and monitor data flow and receive commands from readout computers; to delivery sub-event packages to readout computer.

THE FRAMEWORK OF THE READOUT TEST MODEL

Fig.2 illustrates a prototype of one readout branch of the readout subsystem. Host is the SUN Ultra10 workstation running Solaris 2.6 operating system, in which integrated development environment Tornado2.0 and some development tools should be installed. Target CPU is a single-board computer MVME5100 (450 MHz PowerPC750 with 512M SDRAM) on which VxWorks is to run. With Tornado, we can use the cross-development host to manage project files, to edit, compile, link, and store real-time code, to configure the VxWorks operating system, as well as to run and debug codes on the target while under host system control. The readout computer with Pentium4CPU1.6GHz and gigabit Ethernet card runs Linux operating system.

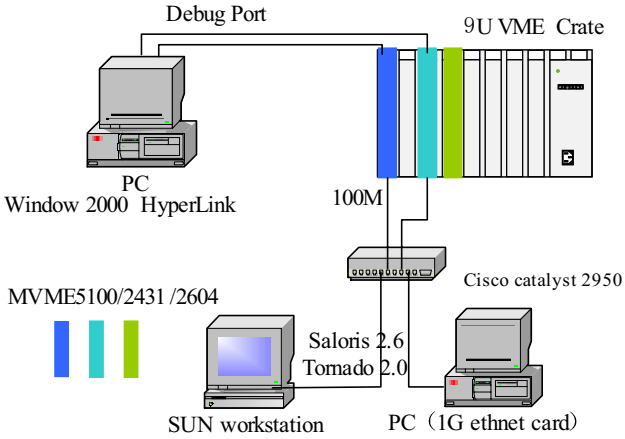

Fig.2 prototype of one readout branch of the readout subsystem

REALIZATION OF THE READOUT TEST MODEL

As shown as Fig.2, The SBC MVME5100 locates in first slot of the VME crate acted as a system controller; The SBC MVME2431 imitates the readout control module which is placed in the third slot, and always runs the procedure to generate interrupts; the SBC MVME2604 was used as the FEE readout module that located in the fourth slot. The SBC will start a DMA read using 32-bit block transfer mode as soon as it responds the interrupt from MVME2431, reading out the data which has been written into one certain memory space of the MVME2604. When the DMA transfer is finished, the data will be temporarily stored in the ring buffer of the SBC MVME5100, then assembled and written into another ring buffer in the mean time. Finally, they will be sent to the readout computer via fast Ethernet network.

In this case, four tasks were spawned in the same main routine: transferring data with DMA mode (called Tdma); unpacking and packing data into a tagged data package (called Tpack); delivering data to readout computer via fast Ethernet (called Tnet); calculating idle time (called Tidle). Each spawn created a new task with its own stack and context. All applications were organized into independent, though cooperating programs. Semaphores were used to realize the fastest intertask communication and task synchronization. With a pre-emptive priority-based scheduler provided by VxWorks, each task had a priority and the kernel ensured the CPU was allocated to the highest priority task that was to run. In the four tasks, Tdma possessed of the highest priority to respond the interrupt and start dma, and Tnet have priority over Tpack and Tpack over Tidle. Two 5M bytes ring buffers were used to store the data temporarily in order that all the tasks can run concurrently; which one storing unpacked data, and another one storing packed data temporally. After the Tdma had been started, the processor was then free to run some other lower priority tasks. Tpack had a lot of arithmetic operation, shift operation and memory-access operation and so on so that it would cost more CPU overhead. To avoid frequent overflow of the ring buffer, Tnet have priority over Tpack. This improves the utilization of the processing resource.

THE KEY FEATURES

Interrupt, Memory-access and DMA

All Motorola VME CPUs use the same industry standard PCI bus to VME bus bridge chip (Universe). This chip provides 64-bit PCI bus interface and VME bus interface, programmable DMA controller for high-performance data transfer between the PCI bus and VME bus, wide range of VME bus address and data transfer modes, interrupt controller and full VME bus system controller functionality also. As a DMA controller, it is operated through a series of registers that control the source and destination for the data, length of the transfer protocol to be used. There are two modes of operation for the DMA: Direct Mode and Linked list Mode. In direct mode, we must know the address of all modules in every one crates and the data length before DMA. Unlike direct mode, in which the DMA performs a single block of data at a time, linked-list mode allows the DMA to transfer a series of non-contiguous blocks of data without software intervention. Each entry in the linked-list is described by a command packet which parallels the DMA register layout.

Multi-cast (MCST) and Chained block transfer (CBLT) mode

As two new standard protocols, Multi-cast (MCST) and Chained block transfer (CBLT) enable a master to address many slaves at once. From the Master's point of view, the CBLT is a normal BLT or MBLT. The master has only to handle the bus error in a correct and fast way. Generally, the bus error signal is asserted by the last slave after all the data is read out for an event at the end of a CBLT. In the readout system applications, the CBLT method will be adopted. In CBLT mode, more CPU time and total time are saved for reading no transfer size and only one same address of the readout modules before starting a DMA.

Intertask communication and synchronization

Semaphores are highly optimized and provide the fastest intertask communication mechanism in VxWorks. For task synchronization, two binary semaphores respectively coordinate Tdma's and Tidle's execution with external interrupt. In addition, two counting semaphore is used for task synchronization, which represent a condition that a task is waiting for. Initially the semaphore is unavailable (empty). Fig 3 illustrates tasks synchronization and the flow chart of three tasks.

Ring buffer

Two specialized ring buffer that can be used for DMA or CBLT are created and used here, and they are different from ring buffer provided by VxWorks library. In particular, the memory block used in Tdma and Tnet needs to be continuous, and the cache flushing and invalidation operations should be used to keeps cache coherency. Besides, a mutual-exclusion semaphore guarding mechanism is used in the ring buffer applications, in such a way, simultaneously accessing by multiple readers and writers would be interlocked.

Cache coherence

If buffers are marked cacheable, DMA device access with caches must guarantee cache coherency, which means the new data in the cache must be coherent with data in RAM. If a CPU writes data to RAM that is destined for a DMA device, the data can first be written to the data cache, When the DMA

device transfer the data from RAM, but the data output to the device may not be updated with the data in the cache. This data incoherency can be solved by making sure the data cache is flushed to RAM before the data is transferred to the DMA device. In the same way, if a CPU reads data from RAM that originated from a DMA device, the data read can be from the cache buffer and not the data just transferred from the device to RAM. The solution to this data incoherency is to make sure that the cache buffer is marked invalid so that the data is read from RAM and not from the cache.

Figure 3: illustration of tasks synchronization and the flow chart of three tasks.

Handshake for interrupt

To imitate MVME2431 with L1 trigger, so that MVME2431 generate an interrupt when event data is ready, then MVME5100 responds it and read out all this data at once. In order to guarantee the continuity of event data received by MVME5100, no interrupt should be lost. That is to say, MVME5100 can just enable the next interrupt sent by MVME2431 after both responding the last interrupt and finishing a DMA transfers.

VxWorks configuration

For VxWorks is a flexible, scalable operating system with numerous facilities that can be tuned, and included or excluded, we need a customized VxWorks kernel with essential function.

THE ANALYSIS OF THE RESULT

So far, we took a series of tests for one crate's data size of EMC and got some results. According to the requirement of EMC, there are no more than 16 Q modules in each crate and on each module there have 32 channels. The hit data should be stored in the global

buffer on board and be read out by the SBC MVME5100. Assuming 17% occupancy for EMC, there would be 32×17%≈6 word (32bits) on one module for each trigger. At both the beginning and the ending of the data of each trigger, a 32bit header and a 32bit trailer would be generated. According to 4 KHz event rate, the data size of one crate is up to 4K×16×(1+32×17%+1) ×4 ≈2048K (bytes/S) and the network throughput is up to 1158K (bytes/S) after data assembling. In the testing environment mentioned above, we got the time performance for tasks running individually and cooperatively which illustrated in Fig 4. In order to get the CPU occupancy, all tasks running with the greatest extreme speed. Accordingly, the CPU occupancy for handling 4000 event data that shown in Fig 4.

The procedure of data assembling adopts two different algorithms, one method to get the detector physics address and the calibrated data via checking one array and another one with no checking. In integrated tests, the more event data were read out for one trigger interrupt, the less CPU occupancy was consumed, since more CPU time being saved by Tdma.

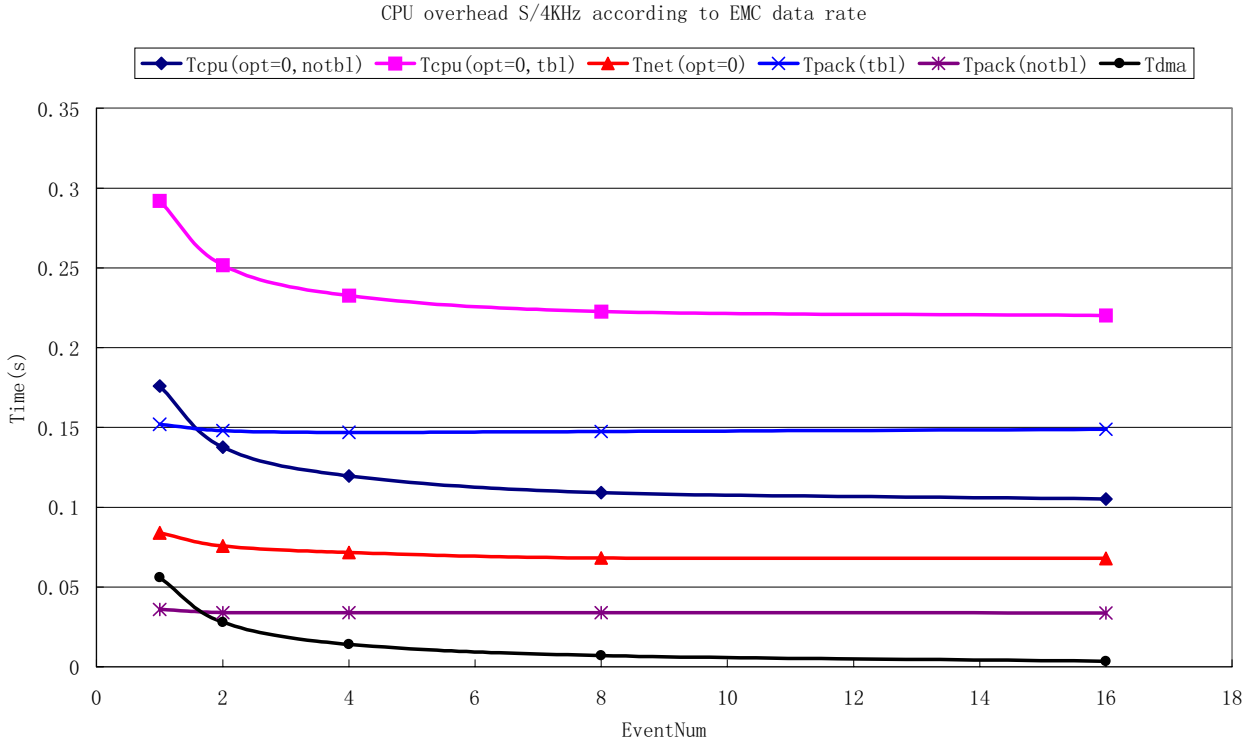

Figure 4: CPU overhead S/4KHz according to EMC data rate

CONCLUSIONS

In summary, the read out subsystem plays an important role in the BESIII DAQ system. To validate the performance of readout subsystem, this readout test model was built and finally realized data collecting, pre-processing and transferring from the VME crate to the readout computer via LAN. It also took into account some key technologies in design. Furthermore, multi-processors can be applied to one crate for taking larger amount of event data and increasing more CPU redundancy. The current prototype of the readout subsystem supplies feasible experimental evidences and references to achieving the real system.

REFERENCES

[1] BES collaboration, BESIII Technical Design Report, Ver1, IHEP 2001.

[2] MVME5100 Single Board Computer Programmer's Reference Guide, V5100A/PG3, July 2003 Edition.

[3] Universe IITM User Manual, Tundra, spring 1998

[4] American National Standard for VME64 Extension for Physics and Other Applications, ANSI/VITA 23-1998

[5] particle detector and DAQ, First Edition, Xie YiGang etc. China Science Press, 2003