General transmission method based on PXI platform for physical experiments

YAO Lin[1,2] CAO Ping[1,2,*] HUANG Xiru[1, 2] SONG Kezhu[1 2] AN Qi[1,2]

[1]*State Key Laboratory of Particle Detection and Electronics (IHEP-USTC), Hefei 230026, China*

[2]*Department of Modern Physics, University of Science and Technology of China, Hefei 230026, China*

Abstract In this paper, a general method of data transmission system design on PXI platform is proposed. It can be used in readout system design for physical experiments. It aims at providing reusable and general interfaces for customized design of PXI while maintaining the transmission performance. It has three main features: (1)universal logic hardware interface, (2)ethernet based socket software interface, and (3)specific and simple data transmission protocol. Data transmission on PXI bus can be realized with the said two universal interfaces coordinated by this specific protocol. Test shows that this method is feasible and stable. This method can be easily reused in readout system designs for different experiments.

Key words General Transmission Method, PXI, FPGA, Ethernet Socket

1 Introduction

Nowadays, the readout system design for modern physical experiments faces the improving requirements of flexible architecture, convenient management, high availability and expansibility at high data throughput. The legacy VME based readout system design methodology is facing more and more difficulties with these requirements. It is limited bandwidth and lagging bus architecture prevents it moves to the road of high data throughput and availability. The lack of hot-swap supporting also makes it difficult to meet the requirement of keeping online at all-time without interruption in a week. Therefore, other high-speed transmission platforms like Compact PCI, PCI eXtension for Instruments (PXI) and Advanced Telecommunications Computing Architecture (ATCA) have been considered as replacement of VME. Compact PCI is ruled out for its lack of dedicated timing or trigger signals. The attractive features of ATCA include high transmission bandwidth, data flexible exchange and high availability. However, its transplanting to physics is still under construction[1]. The PXI provides integrated timing and synchronization module to route synchronization clocks and triggers which matches the requirement of most experiments quite well. It supports a maximum data transmission bandwidth of 528 MB/s with 64 bit data width at 66 MHz frequency while keeping high availability and low cost[2]. For these reasons PXI is utilized in more and more newly constructed physical experiments such as the central timing system of EAST[3], the real time data acquisition system of TJ-II device[4] and the real time data acquisition for the HICAT facility[5]. Some other experiments have adopted PXI to replace the use of VME for upgrade, such as the DAQ system upgrade of MAST [6], and the MHD control system of FTU[7].

There are many commercial PXI modules which can be used directly in readout system design of physical experiments[4,6]. The commercial modules have high convenience but low flexibility due to the mass production. It may not ensure full satisfaction when facing some specific demands. To meet the various needs, many scientists start to customize PXI modules as what they have done under the VME

Supported by National Natural Science Foundation of China (NSFC) projects (Nos.11035001,11120101005 and 11175085) and the Priority Academic Development of Jiangsu Higher education Institutions.

* Corresponding author. *E-mail address*: cping@ustc.edu.cn

Received date: 2013-01-10

platform before[3,7]. The customized module could be developed aiming at the exact features of experiment and could be optimized to obtain higher efficiency. However, this customized design methodology raises new problems such as the requirement of longer developing time and higher cost. In addition, customized modules are always too dedicated to be reused in other applications.

To overcome the disadvantages of customized design method, a PXI platform based general data transmission method is proposed in this paper. It provides a universal data transmission interface to connect heterogeneous data sources from the front-end electronics and DAQ. The universal interface consists of hardware interface and software interface. The hardware part is FPGA internal logic port interface and the software part is Ethernet socket based interface[8]. To coordinate the multiple data source transmission, a specific protocol is also proposed. Controlled by this protocol, the packaged data fed to a port of hardware interface can be received at the corresponding port of software interface. Based on this strategy, designers only need to concentrate on the system function or data processing implementation without considering the detail of data transmission.

2 System architecture

This method is aiming at developing PXI based general and reusable data transmission interfaces while maintaining the transmission performance. As shown in Fig.1, it is realized in middleware form, including hardware middleware and software middleware. The hardware middleware provides a universal logic port interface and the software middleware provides Ethernet based socket port interface. Each functional hardware module has an individual corresponding socket port to control and obtain data. A maximum amount of 255 hardware modules can be supported by this method. Local area readout system can obtain data directly through the socket port[9]. Furthermore, long distance Ethernet communication technologies can be imported to realize remote readout system[10,11].

The PXI bus is a bidirectional bus. In this paper, the data transmitted from software middleware to hardware middleware is called downlink data. The data transmitted from hardware middleware to software middleware is called uplink data. The system is operating based on separated acknowledgements of the two directional data transmission request. Fig. 2 shows the system's state diagram.

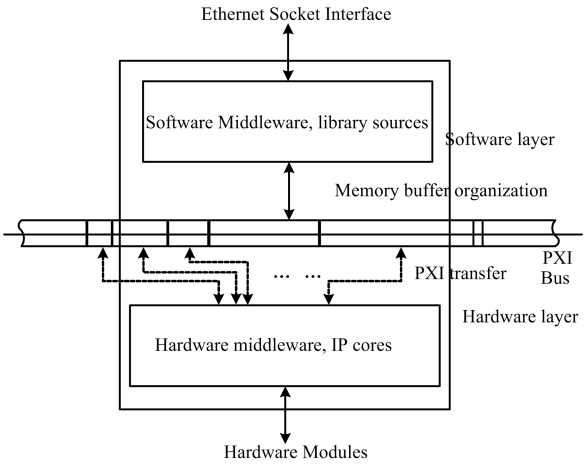

Fig.1 System architecture.

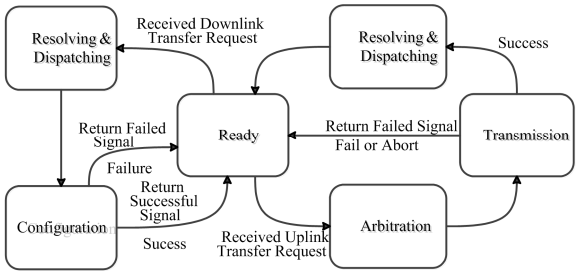

Fig.2 System state diagram.

The system will begin at Ready state when power on. When receiving downlink data transfer request, it resolves the data frame, gets the identification address of target module and moves to Configuration state. Regarding to the configuration status, a successful or failed signal will be returned. When receiving an uplink transfer request, the system steps in Arbitration state to decide which hardware module could start data transmission. If the data transmission is unsuccessful, it will return a failed signal. When the data transmission finishes successfully, the software middleware will resolve and dispatch the data to corresponding socket port. Detailed implementation will be introduced in the following paragraphs.

3 Implementation

3.1 System protocol implementation

To ensure the system could accommodate the various data formats from different hardware modules, a general and simple protocol is proposed. It restricts the transmission data format, and all data must be packaged under this protocol before sent to the provided interfaces. PXI has two data transfer mode: single transfer and burst transfer. All data transmission activities within this system are abstracted as one of these two modes. The single transfer mode is used to transfer the downlink data for configuration or control. It is also used to transfer the slow rate uplink data, e.g., the feedback status or temperature data. The burst transfer mode is used to transfer the large amount uplink digitalized data.

This protocol defines data frame format for the two different modes. A 1 Byte address code is defined to identify the different hardware modules. The address code 0x00 represents the control module of this system and other codes from 0x01 to 0xff can be used to represent up to 255 different modules, which is the nominal module support number of this method. For single transfer mode, the data frame length is 4 byte, 1 byte for address code and 3 bytes for valid data. For burst transfer mode, the data frame length is 1028 byte with a 4 byte header and following 1024 bytes valid data. The 4 byte header consists of 1 byte address code and other 3 bytes reserved parameters. The data frame format is shown as Fig.3 below.

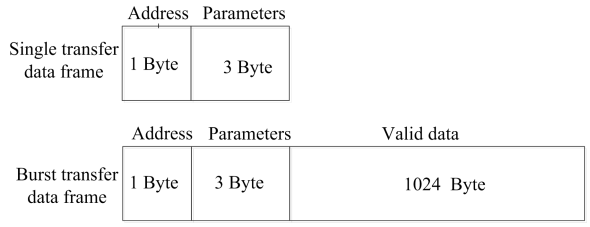

Fig.3 Data frame format.

All data should be packaged under the protocol, transmitted to the provided interface and can be obtained from the corresponding interface from the other side. The PXI bus transmission can be automatically completed by the middleware. The detailed realization of hardware and software middleware is presented.

3.2 Hardware middleware implementation

The hardware middleware is built in a Field Programmable Gate Array (FPGA). It is the bridge of the PXI bus and the hardware modules. It receives the downlink data from the software middleware via PXI bus and dispatches it to correct hardware module. It also receives the uplink data from the hardware modules and forwards it to the software middleware. The PXI bus transmission is controlled with the help of PXI interface logic. A FPGA internal on-chip bus with corresponding bus arbitrator and controller is designed to accommodate the data transmission of various hardware modules. A universal logic port interface is provided to each different hardware modules. The architecture view of hardware middleware is shown as Fig.4.

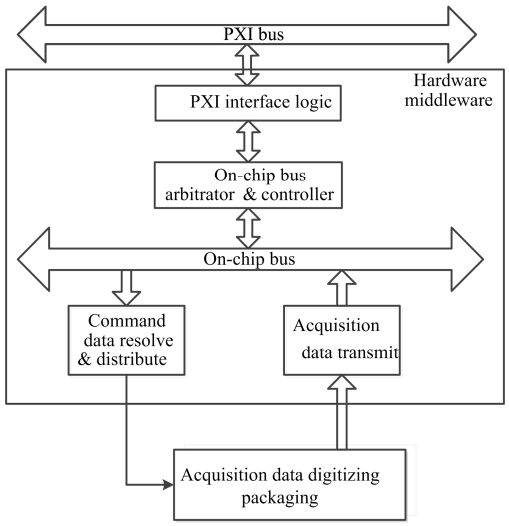

Fig.4 Hardware middleware structure.

The PXI Interface Logic is developed based on the commercial Intellectual property (IP) core integrated in FPGA. The different hardware modules are mounted on the on-chip bus via the provided universal logic port interface. The on-chip bus is designed in master-slave structure and the bus controller is the only master unit while all the mounted logic port interface modules working as slave units. Each slave unit must send request before starting transmission with the master unit and only one slave unit can occupy the on-chip bus to communicate with the master unit at a time. The on-chip bus contains a 32 bit bidirectional data bus and a dedicated 8 bit address bus which can support 255 slave units at most.

The logic ports diagram of the slave unit is shown as Fig.5 below. It provides separated logic ports and two clock input ports to communicate with the on-chip bus and hardware modules in different clocks. A First-In First-Out (FIFO) buffer is applied to deal with the cross-clock domain communication.

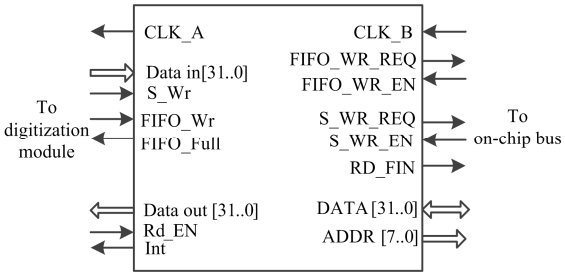

Fig.5 Diagram of logic port interface.

The logic port interface module provides separated uplink and downlink 32 bit data bus to the hardware module with corresponding control ports as shown in Fig. 5. The uplink data transfer has two modes, single transfer and burst transfer. The two modes have the same data port but different control ports. While conflicting, the single transfer mode has higher priority.

The logic port interface module communicates with on-chip bus via 32bit bidirectional data bus and 8bit dedicated address bus. It also provides several control ports. To deal with the transmission competition of different modules, a bus arbitrator is developed. Each slave module has its individual request and enable signal to achieve time sharing of bus operation, as is shown in Fig.6.

The on-chip bus is a bidirectional bus with 3 different transfer activities in different priorities. The downlink data transfer has the highest priority, then the uplink data single transfer and at last the uplink data burst transfer. Each slave module has its priority value while initializing. The direction of bus transfer is determined by the address bus. When the value of address bus is 0x00, it means the bus is occupied by one slave to transfer the uplink data. In other situation it means the bus is transferring downlink data to the slave whose address code matches current address bus value. When transferring downlink data, the bus controller resolves the data frame and fetch the address code to set the address bus to inform the target slave unit. The target slave unit should return a read finished signal to inform the bus transfer operation has finished correctly. If there's no such a reply in 8 clock cycles, the controller will recognize it as a fault, return an error signal to the software middleware and clear the address bus.

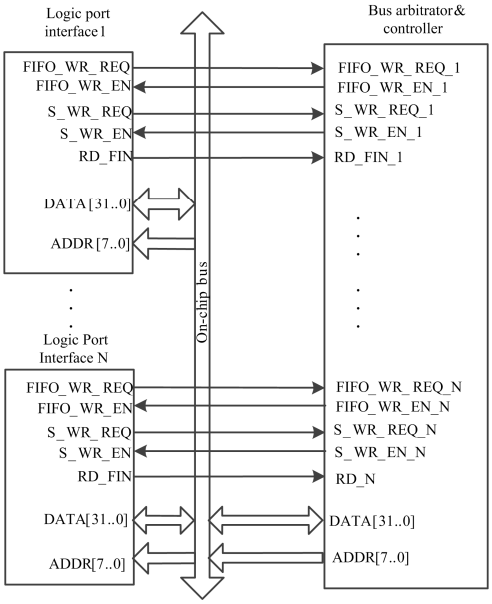

Fig. 6 On-chip bus connection relationship.

When a hardware module needs to start uplink data single transfer, it will send a request signal via the S_WR_REQ port. The bus arbitrator and controller will acknowledge the request if the bus is free and there's no downlink data transfer request. If some else module sent the same request at the same time, the arbitration is carried out. Each module will receive its priority value when system initializing. It can be configured by user's will. If the user didn't configure the priority value at initialization, the default value is the same as module address code. The arbitrator and the controller module will judge the priority of requesting modules and return the S_WR_EN signal of the highest priority module to allow it to start the bus operation.

The uplink data burst transfer is quite similar but it has the lowest priority than the other two modes. The burst transfer request will only be acknowledged when there's no other transfer request. If there is more than one module sending the request, the arbitration starts. It is also determined by the priority value of each module. When received the enable signal, the chosen module occupies the on-chip bus and starts to transfer the 1028 bytes long data frame.

3.3 Software middleware implementation

The structure view of software middleware is show as Fig.7 below. It consists of socket ports, data dispatcher and device driver. The software interface is the socket port which provides Ethernet communication method and supports both local and remote access. All data communicating via the socket port should follow the protocol mentioned above.

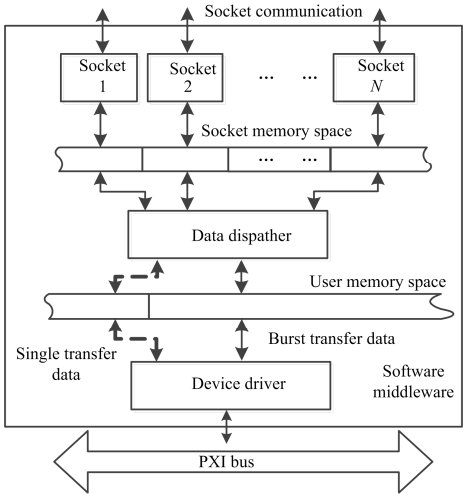

Fig.7 Software middleware structure.

Each socket has its individual memory space to communicate with data dispatcher. The data dispatcher resolves the uplink data frame and fetch the address header to forward the data to corresponding socket memory space. The data in single transfer mode and in burst transfer mode is mapping to two different memory addresses to communicate between data dispatcher and device driver. The device driver realizes software side PXI bus transmission control.

4 Tests

To validate the feasibility and reliability, three PXI 3U boards with 8 different hardware functional modules are developed based on this method. They are:

A) A Dual channel digitization board. The sampling rate of each channel is 80 MSPS in 14 bit.

B) A multifunction board including a 40 MSPS 12 bit ADC, a random signal generator at 10 MHz frequency and a scaler at maximum count rate of 10MHz.

C) A multifunction board including dual channel 12 bit TDC with 1ns resolution and a 250 M 8 bit ADC.

The verification platform is on a 32 bit PXI chassis with 8 slots. The maximum system bandwidth is 132 MB/s. The PXI chassis controller is PXI-8108 and the operating system running on this controller is Windows XP[12].

An experiment is carried out for verification. These three boards are plugged into the chassis. The system tests are conducted in the laboratory environment, with an environment temperature around 26ºC (controlled by the air conditioner). Since the data sampling rate of these module boards far exceeds the nominal bandwidth of this crate, a software timer is adopted to send the cyclical start and stop command to control the sampling operation. In every minute all the sampling modules will receive the start commands. And 1 second later, the stop commands will be sent. The sampled data (about 4.72 Giga bits, which is equivalent to 590 Mega Bytes) will be transferred under our method and in every data frame a single bit even parity code is adopted. If any bit error occurs, a corresponding record will be generated. After the 24 h test, no bit error record is observed.

The system data transmission rate is measured under the following 3 conditions:

a) 80 MSPS single channel in test board A is sampling without this readout method.

b) 80 MSPS single channel in test board A is sampling with this readout method applied.

c) All the 3 boards are working with this readout method applied.

The transmission rate of each situation is measured in 10 times, and the results are shown in the following table.

The nominal bandwidth of PXI is a theoretical one, which is the result of multiplying the bit width by the system clock frequency. In the actual data transmission situation, the transmission speed is mostly limited by the bus scheduling operation. When one slave device sends a request to conquer the bus for transmission, it should wait for the permission from the master device to start it. The permission is usually generated by the operation systems and a time period is always needed. This time period is a very important factor which leads to the difference between the measured transmission rate and the nominal one. This tendency could be seen in test under condition c.

Because of the competition between the modules, the transmission speed is less than the tests under condition a and b but its average speed is still more than 45 MB/s. With this transmission rate it could meet data transmission requirement of experiments such as Daya Bay reactor neutrino experiment[13].

Table 1 Data transmission rated measured in different conditions

Tests	1	2	3	4	5	6	7	8	9	10
a	51.21	50.19	47.46	49.82	49.59	51.13	51.22	49.01	47.14	49.32
b	48.56	47.35	49.34	50.14	49.29	49.16	50.26	48.38	47.67	48.26
c	45.71	46.43	45.18	45.45	46.70	44.81	47.61	44.86	47.80	46.12

5 Conclusion

In this paper, a PXI based general data transmission method is presented. It provides general interfaces and corresponding protocol to help customized design of PXI data readout systems. This method is proposed based on 32 bit PXI bus. The performance of this method is measured and it proves this method to be feasible, effective and easy to use.

When facing higher transmission rate requirement, this method can be considered to be transplanted on other bus platform with higher bandwidth. The 64 bit PXI bus standard can support a maximum 528 MB/s bandwidth and the PXI Express standard can support a much higher 5 GB/s bandwidth. Modifying and transplanting this method to these platforms is a potential further research field and is very possible to obtain higher efficiency.

References

1. Larsen R S. Advances in developing next-generation electronics standards for physics, SLAC-PUB-13684, June 2009.
2. PXI systems alliance, PXI-1 (PCI eXtensions for Instrumentation) hardware specification revision 2.2, 2004, http://www.pxisa.org.
3. Zhang Z, Ji Z, Xiao B, *et al.* Comput Measure Control, 2011, **19:** 2241−2244.
4. Ruiz M, Barrera E, Lopez S, *et al*. Fusion Eng Des, 2004, **71:** 135−140.
5. Peters A, Hoffmann T, Schiwickert M. Beam diagnostic devices and data acquisition for the HICAT facility, Proceedings of DIPAC, 2005.
6. Shibaev S, Counsell G, Cunningham G, *et al*. Fusion Eng Des, 81 2006, **81:** 1789−1793.
7. D'Antona G, Cirant S, Davoudi M. IEEE Tns Nucl Sci, 2011, **58:** 1503−151.
8. Austin Joint Working Group, Portable Operating System Interface (POSIX(R)), 1003.1−2008-IEEE Standard for Information Technology, http://standards.ieee.org/findstds/standard/1003.1−2008.html.
9. Rodrigues S, Anderson T E, Culler D E. High-performance local area communication with Fast Sockets, Proceedings of the USENIX 1997 annual technical conference, January 6−10, 1997.
10. Nagai M, Ishidoshiro K, Higuchi T, *et al*. J Low Temperature phys, 2012, **167:** 689−694.
11. Bauer G, Boyer V, Branson J. *et al*. IEEE Tns Nucl Sci, 2008, **55:** 198−202.
12. PXI 8108 specification, http://sine.ni.com/ds/app/doc/p/id/ds-273 /lang/zhs.
13. Li F, Ji X, X. Li, *et al*. IEEE Tns Nucl Sci, 2011, **58:** 1723−1727.