General transmission method based on PXI platform for physical experiments (Postprint)

Authors: YAO Lin, CAO Ping, Xiru Huang, SONG Kezhu, AN Qi

Date: 2023-06-18T00:00:00+00:00

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.

Full Text

Preamble

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
This paper proposes a general method for designing data transmission systems on the PXI platform for physical experiment readout systems. The method aims to provide reusable and general interfaces for customized PXI designs while maintaining transmission performance. It features three main components: (1) a universal logic hardware interface, (2) an Ethernet-based socket software interface, and (3) a specific yet simple data transmission protocol. Data transmission

on the PXI bus is realized through coordination between the two universal interfaces via this protocol. Tests demonstrate that the method is both feasible and stable, and can be easily reused in readout system designs for different experiments.

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

Introduction

The PXI (PCI eXtensions for Instrumentation) platform provides integrated timing, routing, synchronization clocks, and triggers that match the requirements of most physical experiments quite well. It supports a maximum data transmission bandwidth of 528 MB/s with 64-bit data width at 66 MHz frequency while offering high availability and low cost. For these reasons, PXI is being utilized in an increasing number of newly constructed physical experiments, such as the central timing system of EAST, the time data acquisition system of the TJ-II device, and the real-time data acquisition for the HICAT facility. Other experiments have adopted PXI to replace VME for system upgrades, including the DAQ system upgrade of MAST and the MHD control system of FTU.

Modern physical experiments impose increasingly demanding requirements on readout system design, including flexible architecture, convenient management, high availability, and expansibility at high data throughput. The legacy VME-based readout system design methodology faces growing difficulties in meeting these requirements. Its limited bandwidth and lagging bus architecture prevent it from achieving high data throughput and availability. Additionally, the lack of hot-swap support makes it difficult to maintain continuous online operation without interruption.

Consequently, other high-speed transmission platforms such as CompactPCI, PXI, and Advanced Telecommunications Computing Architecture (ATCA) have been considered as VME replacements. CompactPCI is ruled out due to its lack of dedicated timing or trigger signals. While ATCA offers attractive features including high transmission bandwidth and data availability, its adoption in physics applications is limited by high cost and complexity.

Many commercial PXI modules can be used directly in physical experiment readout systems. However, these commercial modules offer high convenience at the expense of low flexibility due to mass production, and may not fully satisfy specific experimental demands. To meet various needs, many scientists have begun customizing PXI modules, as was previously done with VME platforms. Customized modules can be developed to target exact experimental features and optimized for higher efficiency. However, this customized design methodology introduces new problems, including longer development times, higher costs, and excessive dedication that prevents reuse in other applications.

To overcome these disadvantages of the customized design approach, this paper proposes a PXI platform-based general data transmission method that provides a universal data transmission interface to connect heterogeneous data sources from front-end electronics and DAQ systems. The universal interface consists of a hardware interface (FPGA internal logic port) and a software interface (Ethernet socket-based). To coordinate multiple data source transmissions, a specific protocol is also proposed. Under this protocol, packaged data fed to a hardware interface port can be received at the corresponding software interface port. Based on this strategy, designers need only concentrate on system function or data processing implementation without considering transmission details.

2 System Architecture

This method aims to develop general and reusable data transmission interfaces for PXI platforms while maintaining transmission performance. As shown in Fig. 1, it is implemented in middleware form, comprising both hardware middleware and software middleware. The hardware middleware provides a universal logic port interface, while the software middleware provides an Ethernet-based socket port interface. Each functional hardware module has an individual corresponding socket port for control and data acquisition. This method supports a maximum of 255 hardware modules. Local area readout systems can obtain data directly through socket ports, and long-distance Ethernet communication technologies can be imported to realize remote readout systems.

The PXI bus is bidirectional. In this paper, data transmitted from software middleware to hardware middleware is called downlink data, while data transmitted from hardware middleware to software middleware is called uplink data. The system operates based on separate acknowledgements for the two directional data transmission requests. Fig. 2 [Figure 2: see original paper] shows the system's state diagram.

The system begins in the Ready state upon power-on. When receiving a downlink data transfer request, the system resolves the data frame, obtains the identification address of the target module, and moves to the Configuration state. Regarding the configuration status, a success or failure signal will be returned. When receiving an uplink transfer request, the system steps into the Arbitration state to determine which hardware module may start data transmission. If data transmission is unsuccessful, a failure signal will be returned. When finished successfully, the software middleware will resolve and dispatch the data to the corresponding socket port.

3.1 System Protocol Implementation

To ensure the system can accommodate 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 being sent to the provided interfaces. PXI has two data transfer modes:

single transfer and burst transfer. All data transmission activities within this system are abstracted as one of these two modes. Single transfer mode is used for downlink data transmission for configuration or control, and also for slow-rate uplink data such as feedback status or temperature data. Burst transfer mode is used for large-volume uplink digitized data.

This protocol defines data frame formats for the two different modes. A 1-byte address code identifies different hardware modules. Address code 0x00 represents the control module of this system, while codes from 0x01 to 0xFF can 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 bytes: 1 byte for address code and 3 bytes for valid data. For burst transfer mode, the data frame length is 1028 bytes with a 4-byte header followed by 1024 bytes of valid data. The 4-byte header consists of 1 byte address code and 3 bytes reserved parameters. The data frame format is shown in Fig. 3 [Figure 3: see original paper].

All data should be packaged under the protocol, transmitted to the provided interface, and can be obtained from the corresponding interface on the other side. PXI bus transmission can be automatically completed by the middleware.

3.2 Hardware Middleware Implementation

The hardware middleware is built in a Field Programmable Gate Array (FPGA). It serves as the bridge between the PXI bus and hardware modules, receiving downlink data from software middleware via the PXI bus and dispatching it to the correct hardware module. It also receives uplink data from hardware modules and forwards it to software middleware.

PXI bus transmission is controlled with the help of PXI interface logic. An FPGA internal on-chip bus with corresponding bus arbitrator and controller is designed to accommodate data transmission for various hardware modules. A universal logic port interface is provided to each different hardware module. The architecture of the hardware middleware is shown in Fig. 4 [Figure 4: see original paper].

The PXI Interface Logic is developed based on a commercial Intellectual Property (IP) core integrated in the FPGA. Different hardware modules are mounted on the on-chip bus via the provided universal logic port interface. The on-chip bus uses a master-slave structure, where the designed controller is the only master unit while all mounted logic port interface modules operate as slave units.

Each slave unit must send a 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 that can support up to 255 slave units.

The logic port diagram of the slave unit is shown in Fig. 5 [Figure 5: see

original paper]. It provides separated logic ports and two clock input ports to communicate with the on-chip bus and hardware modules in different clock domains. A First-In First-Out (FIFO) buffer is applied to handle cross-clock domain communication.

The logic port interface module provides separated uplink and downlink 32-bit data buses to the hardware module with corresponding control ports as shown in Fig. 5 [Figure 5: see original paper]. Uplink data transfer has two modes: single transfer and burst transfer. The two modes share the same data port but have different control ports. When conflicts occur, single transfer mode has higher priority.

The logic port interface module communicates with the on-chip bus via a 32-bit bidirectional data bus and 8-bit dedicated address bus. It also provides several control ports. To handle transmission competition among 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 shown in Fig. 6 [Figure 6: see original paper].

The on-chip bus is a bidirectional bus with three different transfer activities at different priorities. Downlink data transfer has the highest priority, followed by uplink data single transfer, and finally uplink data burst transfer. Each slave module has its priority value initialized at system startup, which can be configured by the user. If the user does not configure the priority value at initialization, the default value is the same as the module address code. The arbitrator and controller module will judge the priority of requesting modules and return the S_{{{WR}}_{{EN}}} signal to the highest-priority module to allow it to start bus operation.

Uplink data burst transfer is quite similar but has lower priority than the other two modes. Burst transfer requests will only be acknowledged when there are no other transfer requests. If more than one module sends requests, arbitration begins, also determined by each module's priority value. When the enable signal is received, the chosen module occupies the on-chip bus and starts transferring the 1028-byte data frame.

3.3 Software Middleware Implementation

The structure of the software middleware is shown in Fig. 7 [Figure 7: see original paper]. It consists of socket ports, a data dispatcher, and a device driver. The software interface is the socket port, which provides Ethernet communication capability and supports both local and remote access. All data communicated via the socket port should follow the protocol mentioned above.

Each socket has its individual memory space to communicate with the data dispatcher. The data dispatcher resolves the uplink data frame and extracts the address header to forward the data to the corresponding socket memory space. Data in single transfer mode and burst transfer mode is mapped to two

different memory addresses for communication between the data dispatcher and device driver. The device driver implements software-side PXI bus transmission control.

4 Tests

To validate feasibility and reliability, three PXI 3U boards with eight different hardware functional modules were developed based on this method:

A) A dual-channel digitization board with 80 MSPS sampling rate per channel at 14-bit resolution.

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 10 MHz.

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

The verification platform uses a 32-bit PXI chassis with eight slots, providing a maximum system bandwidth of 132 MB/s. The PXI chassis controller is a PXI-8108 running Windows XP. An experiment was carried out for verification.

The three boards were plugged into the chassis in a laboratory environment with temperature controlled at approximately 26°C by air conditioning. Since the data sampling rate of these module boards far exceeds the nominal bandwidth of this crate, a software timer was adopted to send cyclical start and stop commands to control sampling operation. Every minute, all sampling modules receive start commands, followed by stop commands one second later. The sampled data (about 4.72 Giga bits, equivalent to 590 Mega Bytes) is transferred using our method, with a single-bit even parity code adopted in every data frame. If any bit error occurs, a corresponding record is generated. After 24 hours of testing, no bit error records were observed.

The system data transmission rate was measured under three conditions:
a) 80 MSPS single channel on test board A sampling without this readout method.
b) 80 MSPS single channel on test board A sampling with this readout method applied.
c) All three boards working with this readout method applied.

The transmission rate of each scenario was measured 10 times, with results shown in Table 1 . The nominal bandwidth of PXI is theoretical, calculated by multiplying bit width by system clock frequency. In actual data transmission, performance is primarily limited by bus scheduling operations. When a slave device requests bus access for transmission, it must wait for permission from the master device to start. This permission is usually generated by the operating system, requiring a time period that significantly contributes to the

difference between measured transmission rate and nominal bandwidth. This effect is evident in test condition c, where competition between modules reduces transmission speed compared to conditions a and b, though the average speed remains above 45 MB/s. This transmission rate meets the data transmission requirements of experiments such as the Daya Bay reactor neutrino experiment.

5 Conclusion

This paper presents a PXI-based general data transmission method that provides general interfaces and a corresponding protocol to facilitate customized design of PXI data readout systems. The method is proposed based on the 32-bit PXI bus, and performance measurements prove it to be feasible, effective, and easy to use. For applications facing higher requirements, this method can be considered for transplantation to other bus platforms with higher bandwidth. The 64-bit PXI bus standard supports a maximum bandwidth of 528 MB/s, and the PXI Express standard supports a much higher 5 GB/s bandwidth. Modifying and transplanting this method to these platforms represents a promising research direction with potential for achieving higher efficiency.

References

1. Ruiz M, Barrera E, Lopez S, et al. Fusion Eng Des, 2004, 71: 135−140.

2. Peters A, Hoffmann T, Schiwickert M. Beam diagnostic devices and data acquisition for the HICAT facility, Proceedings of DIPAC, 2005.

3. Shibaev S, Counsell G, Cunningham G, et al. Fusion Eng Des, 2006, 81: 1789−1793.

4. D'Antona G, Cirant S, Davoudi M. IEEE Tns Nucl Sci, 2011, 58: 1503−1511.

5. 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.

6. 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.

7. Nagai M, Ishidoshiro K, Higuchi T, et al. J Low Temperature Phys, 2012, 167: 689−694.

8. Bauer G, Boyer V, Branson J, et al. IEEE Tns Nucl Sci, 2011, 19: 2241−2244.

9. Larsen R S. Advances in developing next-generation electronics standards for physics, SLAC-PUB-13684, June 2009.

10. PXI 8108 specification, http://sine.ni.com/ds/app/doc/p/id/ds-273/lang/zhs.

11. PXI Systems Alliance, PXI-1 (PCI eXtensions for Instrumentation) hardware specification revision 2.2, 2004, http://www.pxisa.org.

12. Li F, Ji X, Li X, et al. IEEE Tns Nucl Sci, 2011, 58: 1723−1727.

13. Zhang Z, Ji Z, Xiao B, et al. Comput Measure Control, 2011, 19: 2241−2244.

Note: Figure translations are in progress. See original paper for figures.