PROCEEDINGS OF SCIENCE

Ethernet-based slow control system for parallel configuration of FPGA-based front-end boards

Wojciech M. Zabolotny*
Institute of Electronic Systems, Warsaw University of Technology
E-mail: wzab@ise.pw.edu.pl

The Ethernet network is a good control interface for distributed measurement systems. The de facto standard in HEP experiments is IPbus. The experiences from using IPbus resulted in the proposal of a new Ethernet-based control interface optimized for quick parallel configuration of multiple systems. The system ensures reliable delivery of control commands and responses. The adverse effects of the Ethernet round-trip latency are minimized by grouping the multiple commands in a single network packet, including the basic handshake operations like waiting with timeout until a specified condition is met. The performance may be increased by using multiple packets "in flight". Usage of Layer 2 Ethernet frames minimizes the FPGA resource consumption. Implementation of the software part in Linux kernel space reduces dependency on specific software packages and libraries.

Topical Workshop on Electronics for Particle Physics (TWEPP2018)
17-21 September 2018
Antwerp, Belgium

*Speaker.

1. Introduction

The idea to use the Ethernet interface to control the measurement systems is well established. Currently, the IPbus [1, 2] is the Open Source standard for such control. It is mature, well tested and used in many applications. However, for certain use-cases the design of IPbus is unnecessarily complicated. Also for certain control operations its performance may be reduced. This work presents E2Bus, an experimental alternative solution aimed at the reduction of resource consumption and improvement of achievable performance.

2. Main IPbus features

For communication with hardware, IPbus uses the User Datagram Protocol (UDP) over IPv4 via 1Gb/s Ethernet link. That allows handling of packets in the user space at the computer side. UDP packets are routable. However, due to possible losses of the packets, this feature is rather not used. The reliable operation of IPbus is assured by the ControlHub implemented in Erlang-[3], which also converts the UDP-based communication into the TCP-based one. That also allows remote access via LAN or WAN (including tunneling if needed). Communication with the user applications written in C++ or Python is provided by the uHAL library. Standard makefiles for IPbus software support Scientific Linux and Centos. Due to compiler and library dependencies, it may be difficult to compile it for another Linux distributions. To increase performance, IPbus allows combining multiple commands in a packet and supports simple Read-Modify-Write operations.

3. Proposed improvements and implementation of E2Bus

The protocol support in the FPGA may be simplified by usage of Layer 2 (L2) Ethernet frames instead of UDP. That requires that the FPGA and the nearest computer system (the "End Controller" (EC)) are in the same L2 network segment, which is also good for security reasons. For best performance, the L2 frames should be processed in the loadable kernel module. That is especially important for quick handling of acknowledgments and retransmission of not confirmed frames which is essential for reliable communication. That approach was successfully used in FADE-10G protocol [4]. Communication between the machine performing the control algorithm (the "User Controller" (UC)) and EC should use the standard and simple TCP-based protocol. The ZeroMQ [5] was chosen for that purpose. Interfacing between the driver and ZeroMQ protocol may be provided by a simple C-based application. That minimalistic approach enables using a very simple Linux embedded system as an EC, instead of a more powerful machine running ControlHub in IPbus. The IPbus concept of sending multiple commands in a single packet to increase performance may also be extended. Performance of typical control algorithms may be significantly improved by performing simple handshake-related operations in the E2Bus IP core (EBC) [6]. The related commands are described in section 3.4. The last improvement is adding of interrupts support. In IPbus the controlling software must poll the hardware. E2Bus removes that limitation. The possible architecture of E2Bus-based control system is shown in Figure 1, and implementation of different components is described in the next subsections.

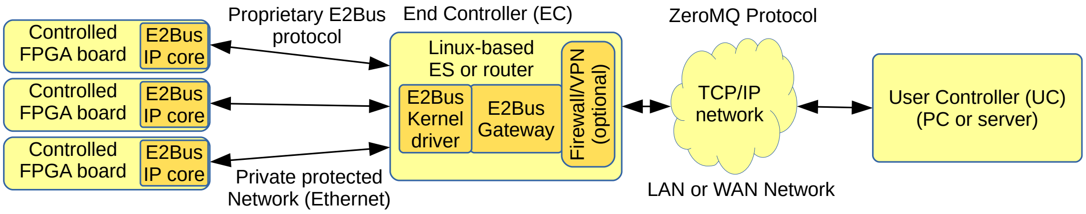

Figure 1: Architecture of the E2Bus based control system.

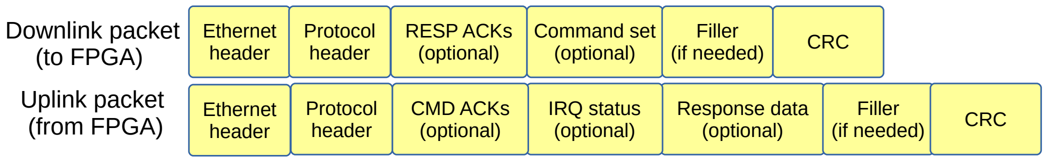

Figure 2: Packets used by the E2Bus protocol.

3.1 E2Bus network protocol

Communication between the EC and controlled FPGA uses L2 Ethernet frames with protocol ID set to 0xe2b5. The layout of the packets is shown in Figure 2. The command sets and response data sets are identified with 15-bit sequence numbers. Different sections of the packets (shown in Figure 2) are uniquely marked so that the presence of the particular section can be easily detected, and its content found. The uplink packets are sent if there is an active interrupt, if a new command set is received, or if there exists any unsent or unconfirmed response set. The delay between consecutive retransmissions and interrupt request (IRQ) notifications is configurable.

3.2 Implementation of End Controller

The EC is a Linux-controlled computer with installed *e2bus.ko* kernel driver used by a simple C-implemented E2Bus gateway (*e2bus_gw*) that provides communication between the UC and EC via the ZeroMQ protocol [5]. If control via a public network is needed, additional encrypted tunnel or VPN may be added. The execution of the commands uses the ZeroMQ PAIR pattern, while notifications about interrupts use the ZeroMQ Publish/Subscribe pattern.

3.3 Implementation of the E2Bus kernel driver

To minimize the acknowledgment and retransmission latency, the kernel driver installs its private protocol handler in the Linux kernel. Communication with the user space application (usually the *e2bus_gw*) is done via ioctl calls. Connection with the FPGA board is started with *E2B_IOC_OPEN* ioctl. It also informs the EBC about the MAC of the EC and resets the EBC (without resetting of controlled peripherals). When the board is connected, the user space application may submit the list of commands for execution. The list should not be longer than 1024 bytes. It may be submitted for synchronous (*E2B_IOC_SEND_SYNC*) or asynchronous (*E2B_IOC_SEND_ASYNC*) execution. The latter allows submission of multiple sets in a sequence for maximal performance. The application may wait until the command set is executed using

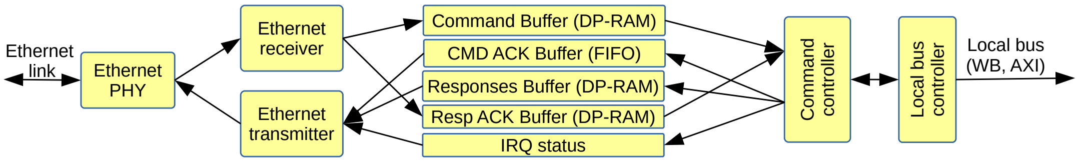

Figure 3: Structure of the E2Bus IP-core.

E2B_IOC_RECEIVE ioctl. The standard poll function is also supported for single-threaded e2bus gateways. To minimize the copying of data, the received response is written to the buffer provided by the user space application and mapped using the *get_user_pages* function after the appropriate *E2B_IOC_SEND_xxxx* call. The ioctl *E2B_IOC_WAIT_IRQ* function allows waiting in a separate thread for interrupts.

3.4 Implementation of E2Bus IP core

The architecture of the EBC responsible for the FPGA side of the E2Bus protocol is shown in Figure 3. The most important part of the solution is the command processor that executes the whole sets of commands and generates the response. When the response is too long to fit in a single packet, the packet is transmitted, and filling of the next packet starts. At the end of the last response packet, there is a status of the operation and the address of the last executed command. In case of an error, the next commands are not executed, and next command sets also are rejected. That error state is cleared only when a command set beginning with a special *ERROR-CLEAR* operation is received. That ensures reliable operation in the mode with multiple packets “in flight”. The Command Controller supports five operations: *READ* and *WRITE* [1], *READ-MODIFY-WRITE* [2], *READ-AND-TEST* [3], and *MULTIPLE-READ-AND-TESTS* [4].

4. Results

The first “proof of the concept” implementation of the proposed E2Bus system has been tested. The E2Bus IP core has been implemented as ISE project for Spartan 6 (2691 LUTs, 9 BRAMs) and as Vivado project for Artix 7 (2829 LUTs, 6 BRAMs) FPGA. Versions for 1 Gb/s and 100 Mb/s have been prepared. The E2Bus kernel driver and E2Bus gateway have been compiled and

[1] *READ* or *WRITE* defines a single or multiple (up to 4096) reads or writes. The read source address and write destination address may be incremented, decremented or constant, the write source address may be constant or incremented. Each read appends the received value to the response. The whole *WRITE* command including data must fit in a single 1024-byte block. The *WRITE* command produces no response.

[2] *READ-MODIFY-WRITE* modifies the contents of a register. Possible operations include: Increment, Decrement, Add, Subtract, And, Or, Xor, Not. The original value may be appended to the response.

[3] *READ-AND-TEST* checks for the condition. Possible tests include: Signed/Unsigned less/greater than, Compare, And/Or with mask and compare. If the condition is not met, the calculated value is written to the response, and an error is generated.

[4] *MULTIPLE-READ-AND-TESTS* checks the same condition as *READ-AND-TEST*, and if it is not met, repeats the test up to the programmed number of times with the programmed delay. If the test is finally passed, the number of repetitions left is written to the response. If not, the value calculated in the last repetition is written to the response, and error is generated.

tested on the Intel x86 and ARM platforms. The sources were converted to Buildroot packages to allow easy testing on embedded platforms. The correct operation of the system with a simple set of Wishbone slaves has been proven. The library for automatic generation of command sets and interpretation of responses (an equivalent of IPbus uHAL library) is still in preparation. E2Bus is released as an Open Source project [7].

5. Conclusions

The proposed E2Bus system may be used to build distributed Ethernet-based control systems for FPGA-based boards. Usage of Layer 2 Ethernet frames simplifies the FPGA IP core. The low-latency reliable transport is implemented in the kernel space using the own packet handler. Implementation of the whole system in the loadable kernel module and in simple ZeroMQ server enables the use of even simple Linux-based routers as End Controller nodes in the system. Parallel control of multiple boards is supported by the following features:

- Multiple packets with subsequent commands sets may be submitted for execution. The next set is processed, while the results produced by the previous ones are transmitted.
- Simple handshake operations, like waiting for certain bits to be set or cleared are executed locally by the E2Bus IP core, without generating additional latency and network traffic. Error conditions stop the execution of the whole submitted set of commands. Detailed information about the error is included in the response.
- Interrupts support reduces the need for polling of the controlled hardware

The proposed solution is still not mature. However, the tests of the first implementation have proven the correctness of the concept. Further tests and cleanup of the code are required.

6. Acknowledgment

Work was supported by statutory funds of Institute of Electronic Systems.

References

[1] C. G. Larrea, K. Harder, D. Newbold, D. Sankey, A. Rose, A. Thea et al., *IPbus: a flexible Ethernet-based control system for xTCA hardware*, *Journal of Instrumentation* **10** (Feb., 2015) C02019–C02019.

[2] R. Frazier, G. Iles, D. Newbold and A. Rose, *Software and firmware for controlling CMS trigger and readout hardware via gigabit Ethernet*, *Physics Procedia* **37** (2012) 1892–1899.

[3] "Erlang programming language." http://www.erlang.org.

[4] W. Zabolotny, *Low latency protocol for transmission of measurement data from FPGA to Linux computer via 10 Gbps Ethernet link*, *Journal of Instrumentation* **10** (July, 2015) T07005–T07005.

[5] "ZeroMQ distributed messaging." http://zeromq.org/.

[6] W. M. Zabołotny, *Improvement of FPGA control via high speed but high latency interfaces*, in *Proc. SPIE* (R. S. Romaniuk, ed.), vol. 9662, (Wilga, Poland), p. 96623G, Sept., 2015. DOI.

[7] "E2Bus control of FPGA-based systems via Ethernet interface." https://github.com/wzab/e2bus.