TESLA Report 2004-11

Prototype Implementation of the Embedded PC Based Control and DAQ Module for TESLA Cavity SIMCON

Piotr Roszkowski[a], Wojciech M. Zabolotny[a], Krzysztof Kierzkowski[b],
Krzysztof Pozniak[a], Ryszard Romaniuk[a], Stefan Simrock[c]

[a]Institute of Electronic Systems, Warsaw University of Technology,
ul. Nowowiejska 15/19, 00-665 Warszawa, Poland
[b]Institute of Experimental Physics, Warsaw University, ul. Hoza 69, 00-681 Warszawa, Poland
[c]Deutsches Elektronen - Synchrotron DESY, Notkestrasse 85, 22607 Hamburg, Germany

ABSTRACT

This paper describes the results obtained with the first prototype of the embedded PC (Axis ETRAX LX MCM) based board for control and data acquisition system, to be used in TESLA controller and simulator (SIMCON).

This board is a result of the new approach to the architecture of measurement systems, where the VME controller is replaced with the embedded PC, to improve the functionality and to reduce cost of the system.

This new concept, described in [1] assures better scalability and testability of the SIMCON board.

Keywords: TESLA, DESY, FPGA, embedded systems, embedded PC, simulator and controller boards

1. INTRODUCTION

The standard interface for control and data acquisition system in HEP experiments is a VME bus. Despite the wide use of this bus, it has several disadvantages. The main problem is that the VME bus controller is unable to service many boards at the same time, which affects the performance. Another shortage of the VME bus is that each rack must be equipped with the bus controller witch increases the cost of the system.

The progress of technology allows putting more intelligence in the SIMCON board, and using other interface, better suitable for communication between such "intelligent boards" than the VME bus.

The main idea is to replace the expensive VME bus controller with onboard controller and use another communication interface (eg. Ethernet) for communication between boards.

Even so VME bus is faster than the Ethernet interface available in most embedded systems (typically 100Mb/s, further decreased by the protocol overhead, in opposite to 40MB/s=320Mb/s for VME with 32-bit data bus),its bandwidth is shared between all boards in rack, while with reasonable network configuration (eg. 100Mb/s hub with the 1Gb/s uplink) the full Ethernet's bandwidth is available for all boards independently. Additionally the failure of a single board in the VME based system may lock bus in the whole rack, while in the Ethernet based solution the hub sufficiently separates boards.

Another advantage of Ethernet interface is its wide use, which results in very low prices of the Ethernet cables, switches and other accessories.

Because Ethernet's effective throughput highly depends on the onboard controller's performance, the best solution is to use a PC-like system to utilize the Ethernet interface optimally. Additionally this high performance can be used to implement more complex algorithms to manage board and to boost the overall system efficiency.

The distributed architecture with multiple onboard controllers, tightly connected with the board's bus, provides also better performance when executing such typical operations like register manipulations (where delay introduced by the

Further author information: (Send correspondence to Piotr Roszkowski)
Piotr Roszkowski: E-mail: proszkow@elka.pw.edu.pl

bus interface or bridge should be as low as possible, and some operations should be located in the "critical sections" with switched off interrupts).

This architecture is also advantageous for data acquisition and processing, because this operations can be performed by many controllers in parallel, and only the preprocessed data may be sent to the managing computer.

2. HARDWARE

2.1. Single chip microcomputer

There are many Embedded PCs available on the market, however considering the power consumption, the size and the performance, finally the Axis ETRAX LX MCM[2] has been chosen. This is an almost full PC class single chip microcomputer (SCM). It integrates in a single chip a 32 bit RISC microprocessor compatible with AXIS CRIS (Code Reduced Instruction Set) architecture, memory and many peripheral devices. Performance of this CPU is equal to 100 MIPS with 20 MHz external clock (internally multiplied by a factor of five). The ETRAX LX MCM chip includes also:

- 10/100 MBit Ethernet controller
- 32-bit external memory and peripheral asynchronous bus
- 4 asynchronous serial ports
- 2 synchronous serial ports
- 2 USB ports
- 2 Parallel ports
- 4 ATA (IDE) ports
- 2 Narrow SCSI ports (or 1 Wide)
- 2 8-bits GPIO ports
- 2 MB of FLASH and 8 MB of SDRAM memory
- Support for additional memories (SDRAM, Flash, EEPROM, SRAM, and others)

Additionally in prototype controller board the SDRAM memory has been externally extended to 16 MB, and the flash memory to 4 MB*. Parallel ports, ATA ports and SCSI ports use the same groups of pins and they can not be enabled simultaneously. If all these ports are disabled, the 32-bit GPIO port is available.

ETRAX chip works under control of the popular Linux OS. This chip supports two main ways of booting the operating system: from the onboard flash memory and from the ethernet network. In the controller the second method is used, because when combined with the NFS mounting of the root file system, it allows to avoid wearing off the FLASH memory.

2.2. The FPGA chip

An additional FPGA[3] (Altera ACEX EP1K100) chip provides a simple interface between onboard controller's external peripherial bus and the board's internal bus. This chip also contains necessary logic and interface to configure all FPGA chips on the board and on its motherboard.

The FPGA chip is connected to ETRAX via a 32-bit data bus. The very convenient feature of the ETRAX chip is a possibility to work with different access cycle length for each connected external device. This functionality is achieved by splitting the ETRAX's address space into 64MB areas. Each area uses it's own external chip select (CS) signal, and it's possible to set different access cycle lengths for different areas. Due to this feature the slowest external devices does not reduce the speed of the whole system. The FPGA chip works with 200ns long access cycle.

The simplest way to configure the onboard FPGA chip is to use the ETRAX chip as a configuration device. Altera chips support two configuration interfaces and both are implemented on the ETRAX board.

2.3. Modularity of the design

To allow easy upgrade, the ETRAX based board controller has been implemented as a doughterboard with standarized external connectors. This solution assures easy replacement of the controller with a newer version.

*Currently also a new version: ETRAX 100LX MCM 4+16 is manufactured and is recommended for new designs. This version is equipped with 16MB of internal SDRAM and 4 MB of internal FLASH

3. BOARD ARCHITECTURE

The block diagram explaining the board architecture is shown in the Figure 1.

3.1. Interfaces

- **Altera Passive Serial Loader (APSL)**
  This interface uses one of the 8-bit GPIO ETRAX's ports - port A, and it is dedicated to configure the FPGA chip connected to the ETRAX external peripheral bus.
  Altera Passive Serial loader contains four output lines:
  - DATA0
  - DCLK
  - nCONFIG
  - SOFTRESET

  and three input lines:
  - nINITDONE
  - nSTATUS
  - CONFDONE

  This interface is a very simple and fast way to configure Altera FPGA chips. However it can only be used for that particular purpose.
- **JTAG**
  This is a powerful interface for boundary-scan testing of digital circuits and for configuring of programmable chips. Altera FPGA chips include dedicated hardware to support JTAG interface, but in ETRAX chip the whole JTAG interface must be implemented in software. One of the ETRAX GPIO ports - port G is used to implement the Altera JTAG interface. The JTAG interface is a serial bus and contains four output lines:
  - TCK
  - TMS
  - TDO
  - TRST

  and one input line: TDI
  This interface can be used to configure and test a FPGA chip connected directly to ETRAX.
- **Altera JTAG**
  This is a JTAG interface implemented in onboard FPGA and can be used by software running on ETRAX to configure and control other FPGA chips on the whole board. This interface is more efficient than the previously described software implemented JTAG interface. Access to this interface is provided by the Internal Interface (described in the next section).
- **The Internal Interface**
  The ETRAX system bus is a little different than the bus implemented on the motherboard. The motherboard uses internally an easy to implement VME-like bus.[4] To enable communication between ETRAX and motherboard's internal bus a simple bridge is implemented in the FPGA chip. The onboard FPGA chip is directly connected to the ETRAX system bus and to the motherboard's Internal Bus (Internal Interface).

4. SOFTWARE

All ETRAX GPIO ports and interfaces are mapped into the ETRAX internal memory and can be easily read and written by a software running in the kernel space. This ability is very useful to develop the Linux device driver and to simplify writing of fast and efficient software. In the C language each memory mapped register can be accessed via a standard pointer.

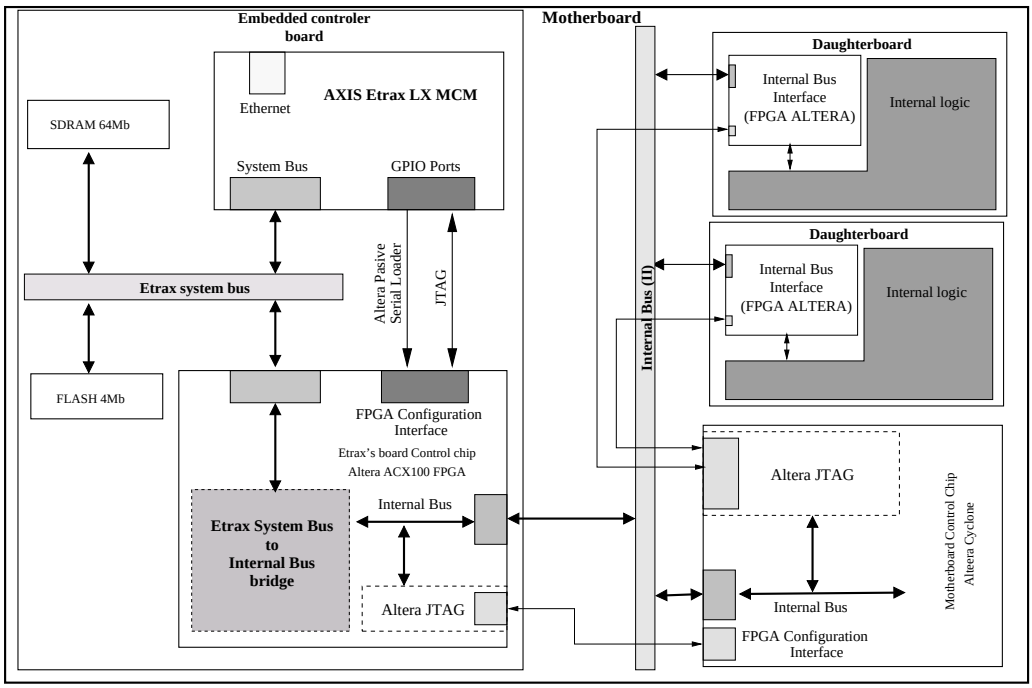

Figure 1. Block diagram of the whole control and data acquisition board.[1]

4.1. Altera Passive Serial Loader (APSL) and JTAG

To reduce the resource usage, these interfaces both use the same Linux Device Driver.

User can select one of this interfaces by opening the corresponding special device file: /dev/altconf for the APSL interface and /dev/jtag for the JTAG interface. Algorithm for the APSL interface is very simple and is completely implemented in the device driver. Therefore no dedicated user space software is needed to service this interface. Because this interface is used just for configuration of the FPGA chip, only the *"write"* function is available for user. User must only write a bit stream configuration file to /dev/altconf file, for example using standard linux "cat" program.

Programing via JTAG interface is realized with the Altera's software called JAM Player. This software contains the full programming algorithm and needs only a simple software interface to drive the JTAG pins. This software interface is activated by the linux device driver when /dev/jtag file is open. Access to the target JTAG lines is realized with the ioctl function. It's possible to change the state of a single JTAG pin or of many JTAG pins simultaneously, it is also possible to read the state of the input JTAG pins. The driver also supports an operation mode with the automatic generation of the clock pulse, whenever the new state of the output pins is set. The ioctl function is called like this
ioctl(file descriptor,COMMAND,ARG | STATE)
where STATE is H or L,
COMMAND is:

- SET_LINE_STATE - set ARG line to state STATE
- GET_TDI_STATE - read state of TDI line
- SET_CLOCK_MODE - set clock mode to AUTO(clock pulse is generated by the device driver) or MANUAL(user can set clock line state using SET_TCK_ command)
- SET_TCK_L - when MANUAL mode is set this command set L state on TCK line.
- SET_TCK_H - when MANUAL mode is set this command set H state on TCK line.
- UPDATE_LINES_STATE - set state for all JTAG lines, state is given as a least significant byte of the ARG.

ARG is:

- TCK - bit mask 0x10
- TDO - bit mask 0x20
- TMS - bit mask 0x80
- TRST - bit mask 0x04

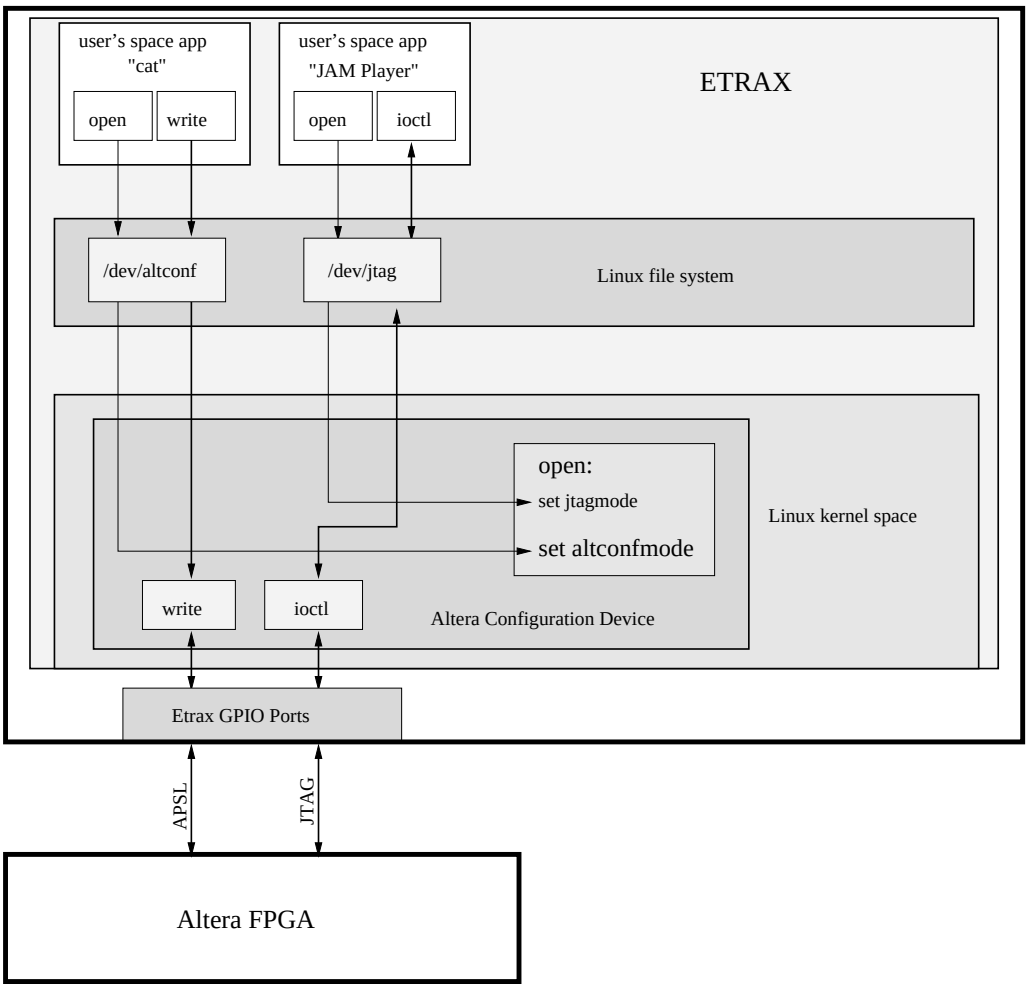

Figure 2. The structure of the APSL and JTAG device driver.

This device driver and user space software (eg. the JAM Player[5]) can be used to configure multiple FPGA chips on board. Only a simple address change is necessary to select the particular chip, because all JTAG interfaces on the board are implemented as registers in the Internal Interface address space, and the programing algorithm is hardware-independent.

4.2. System Bus and Altera JTAG

The board system bus (Internal Interface - II) is connected to the ETRAX bus via hardware bridge implemented in the FPGA chip. From the ETRAX side the IIbus is just one of external devices connected to the system bus. The ETRAX chip reserves 64 MB of the address space for each external device (precisely for each CS signal). Size of the data word in this area is equal to 32 bits. To increase access speed, this area is mapped into the internal ETRAX memory space. Therefore access to the IIbus - accessible registers is similar to access to the internal ETRAX memory and is realized via pointers in C. Additionally standard C structures and bit-fields may be used to describe more complex registers' organization.

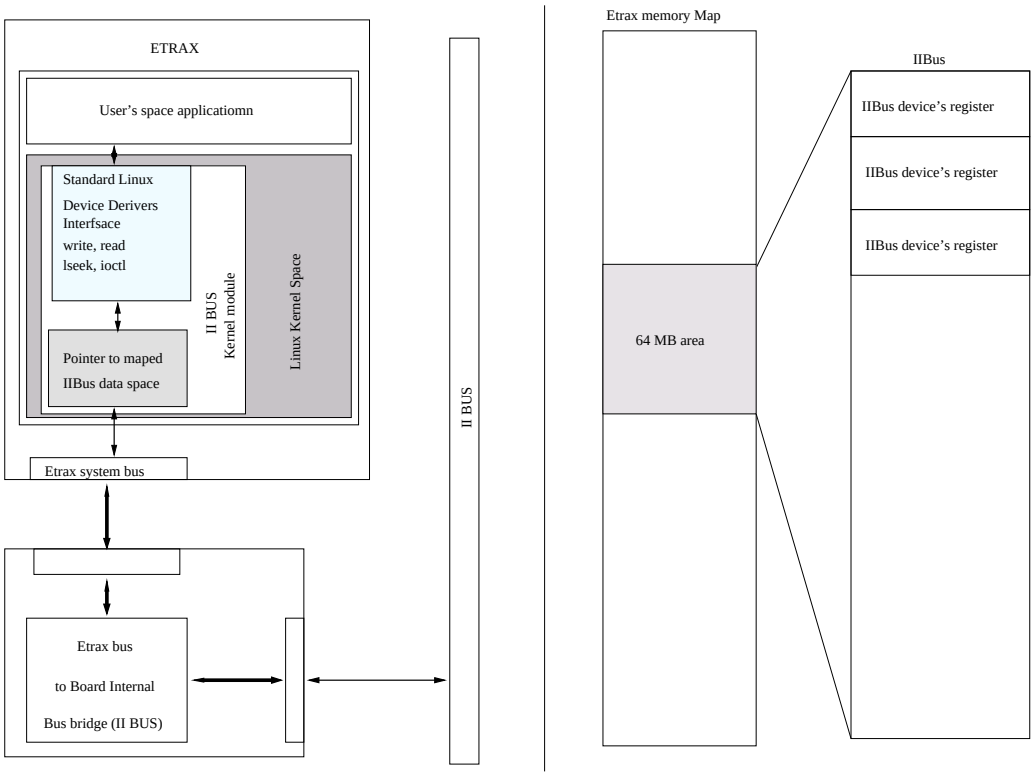

Figure 3. IIbus device driver.

Device driver for IIbus supports 3 types of data word size – 8, 16 and 32 bits. This is necessary because the IIbus architecture allows registers to have different sizes. The character devices use the 8-bit data words, and therefore all 32-bit data must be splited into four 8-bit words. When such words are copied into the kernel space, they are concatenated together to be transferred over the 32-bits wide data bus. When data size was less than 32 bits - the empty bytes should be inserted and transferred between the user and kernel space. To reduce amount of unused data words, the device driver automatically extends data to 32 bits when other data size is set. To set the data word size the *ioctl(int file,COMMAND,ARG)* function is used.

- COMMAND: SET_DATA_SIZE
- ARG: DWORD for 32 bits data; WORD for 16bits data; BYTE for 8bis data

Another feature of this device driver are two addressing modes. The first of them - the BURST mode is used to automatically increment the address, after the start address is set and then data are transferred into/from consecutive address locations. The second one - the SINGLE mode is used to transfer all the data into/from single address location (eg. if it is the address of a FIFO queue).

These two modes are selected with the *ioctl(int file,COMMAND,ARG)* function

- COMMAND: SET_ADDRESS_MODE
- ARG: SINGLE, BURST.

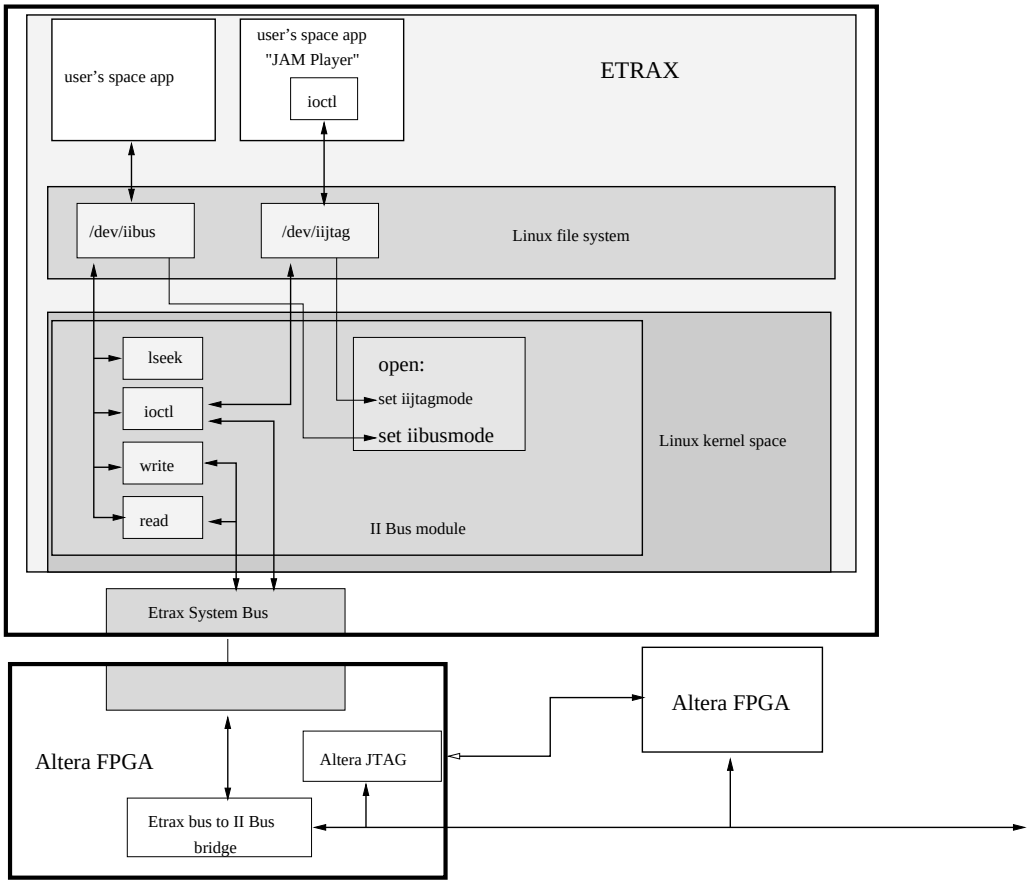

Figure 4. The structure of the IIbus and JTAG device driver.

To set current address the *lseek* function is implemented.

In the Linux user space the access to IIbus is realized as an access to the character device, and standard IO functions like *"write" "read"* and *"lseek"* can be used.

Additionally to reduce the resource usage the IIbus module implements also the Altera JTAG interface. This interface is used to configure the motherboard's FPGA chips and is implemented as a register in the IIbus address space. Software interface for Altera JTAG is provided with the ioctl function in the IIbus module, and is similar to ioctl function in JTAG module for ETRAX JTAG interface.

Access to IIbus is possible via /dev/ii special file, and access to Altera JTAG is possible via /dev/iijtag special file.

Future plans for this device drivers include: development of common device driver for all configuration and communication interfaces, to reduce resource usage and to allow multithread access to the above interfaces with proper synchronization and resources locking.

5. NETWORK PROTOCOLS

Currently the network communication is established with a simple protocol based on TCP connection. This protocol is simple and can be easily implemented in all operating systems supporting TCP sockets.

The protocol uses frames containing: 16-bit id number (constant for all frames), 32-bit IIbus address , 32-bit number of data words, 8 bit information about the address mode and size of data word, and finally the char table containing the data.

This protocol does not support any error control and correction (except the standard mechanisms provided by the TCP/IP stack) and will be replaced witch the RPC based protocol.

6. TEST RESULTS

To check the system performance some simple tests have been made. These tests can be splitted into two categories: FPGA chip programing speed, and the IIbus access speed.

6.1. FPGA programing results

- Altera ACEX EP1K100 on the ETRAX board
  - APSL - 2s
  - JTAG - 20s
- Altera Cyclone on the motherboard:
  - JTAG - 22s

6.2. Test of the transfer rate between ETRAX and IIBus via FPGA chip:

- transfer of 100MB into one address location with correct access test transfer rate with 200ns bus operation time and 800ns delay time - 4MB/s

The Figures 5 and 6 show sample oscilograms recorded during the IIBus operation. Figure 5 shows sample read operation. The upper line shows the state of IIOPER signal which indicates any operation on the IIBus. The lower line corresponds to the IIWRITE signal which determines the type of the operation . High level on this line denotes read operation and low level denotes write operation. This figure also shows the access length equal to 200ns. Figure 6 shows time between two operations on the IIBus. In this oscilogram only IIOPER signal is shown. The time between two operations is equal to 800ns.

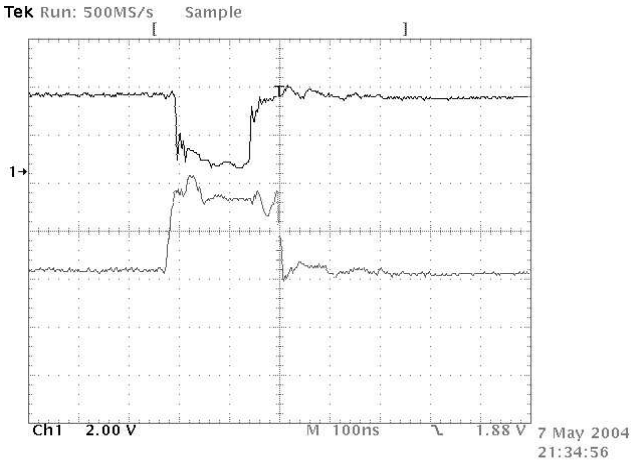

Figure 5. IIbus read operation (200ns operation time). Up - IIOPER line ,down - IIWRITE line

6.3. Test of access speed via the network

- The access speed is limited mainly by transfer rate between ETRAX system bus and FPGA chip, reported in 6.2.

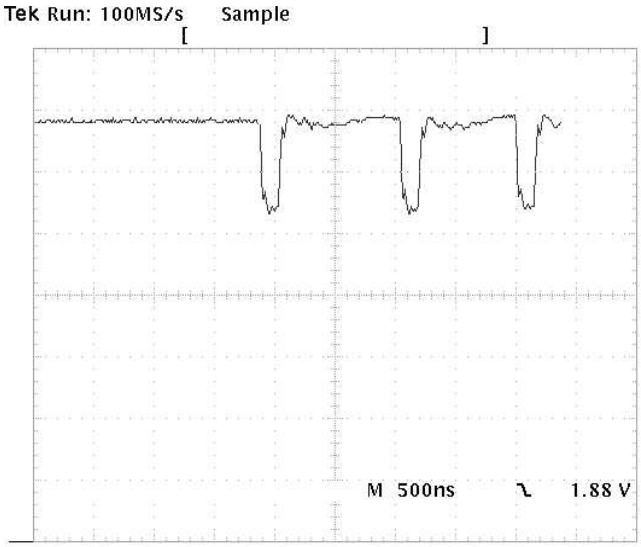

Figure 6. IIbus read operation (800 ns time between two operations).IIOPER line.

The simple comparison of these transfer rates leads to conclusion that the embedded controller's speed is ten times lower than speed of the VME. However VME supports only simultaneous transfers between one VME board and the controller, while in the embedded controllers based system all controllers can communicate with their boards in parallel. Additionally as described before, in the network configuration with 1Gb/s uplink, the maximum speed of the controller's Ethernet network (typically 100Mb/s) is available for each board.

In the present solution the maximum transfer rate is limited by the transfer speed between ETRAX chip and FPGA chip. This transfer speed may be further increased in two ways. The first one - by the use of ETRAX DMA channels. This mechanism should also reduce CPU utilization when data are transfered between ETRAX's memory and FPGA. The second way to increase the transfer rate is to reduce access time on the ETRAX bus. In current implementation 200 ns operation time is used, but future optimization of FPGA code should reduce this time approximately to 100ns. One must also consider that similar limitation affects also the VME transfer rate, so the 320Mb/s is in most cases only a theoretical value).

The use of DMA together with optimization of FPGA code should increase the transfer rate between FPGA and ETRAX chip sufficiently to fully utilize the speed of the Ethernet Network. When used in network with 1Gb/s uplink the resulting overall throughput should be better than in the VME based system.

7. CONCLUSIONS

Typical control and data acquisition system with VME bus has several disadvantages like access for one only board at the same time, and need for expensive VME bus controller. New idea of data acquisition system presented in this note solves these and other problems associated with the VME bus. This new architecture offers also some additional capabilities like implementation of some relatively complex algorithms in onboard chip.

The Ethernet network is a very quickly developed interface. New implementations support greater transfer speed than the VME bus. The modular architecture is very flexible and provides an easy way to replace parts of system with more powerfull ones. Eg. it is possible to replace the controller board with a new one featuring faster Ethernet controller or more powerful embedded PC equipped with more resources like RAM and FLASH memory. These additional resources may be used to implement more complex data processing algorithms, and to shift more data processing tasks from the managing computer to the onboard controller.

REFERENCES

1. W. Zabolotny, K. Pozniak, R. Romaniuk, T. Czarski, I. Kudla, K. Kierzkowski, T. Jezynski, A. Burghardt, and S. Simrock, "Distributed embedded PC based control and data acquisition system for TESLA cavity controller and simulator," *Proceedings of SPIE* **5484**, pp. 223–230, 2004.
2. "Axis etrax LX MCM home page." `http://developer.axis.com/products/mcm/index.html`.
3. "Altera FPGA home page." `http://www.altera.com/products/devices/dev-index.jsp`.
4. K. T. Pozniak, M. Bartoszek, and M. Pietrusinski, "Internal interface for RPC muon trigger electronics at CMS experiment," *Proceedings of SPIE* **5484**, pp. 269–282, 2004.
5. "JAM Player home page." `https://www.altera.com/support/software/download/programming/jam/jam-index.jsp`.

Paper presented during XVIth IEEE-SPIE WILGA Symposium on Electronics & Photonics for HEP Experiments, 26-30 May, 2004, Published in Proc. SPIE