Intelligent Platform Management Controller for Low Level RF Control System ATCA Carrier Board

Predki, Pawel (TUL) *et al*

08 May 2011

The research leading to these results has received funding from the European Commission under the FP7 Research Infrastructures project EuCARD, grant agreement no. 227579.

This work is part of EuCARD Work Package **10: SC RF technology for higher intensity proton accelerators and higher energy electron linacs**.

Intelligent Platform Management Controller for Low Level RF Control System ATCA Carrier Board

Pawel Prędki, Dariusz Makowski, *Member, IEEE*

Abstract—**High availability and reliability are among the most desirable features of control systems in modern High-Energy Physics (HEP) and other big-scale scientific experiments. One of the recent developments that has influenced this field was the emergence of the Advanced Telecommunications Computing Architecture (ATCA).**

Designed for the telecommunications industry it has been successfully applied in other domains such as accelerator control systems. A good example is the application of ATCA standard for the design of Low Level RF (LLRF) control system for the X-Ray Free Electron Laser (XFEL) being developed in Deutsches Elektronen Synchrotron (DESY). Reliability and availability requirements for such a device play a crucial role among other parameters. Thus, the ATCA standard, with five-nines availability, is considered one of the best candidates for this system.

This article focuses on the central management unit of every ATCA board, namely the Intelligent Platform Management Controller (IPMC), developed for the LLRF ATCA Carrier Board (CB). It also argues that it is possible to create a fully functional IPMC using base specifications only which is a much more economical solution than acquiring such products from various vendors dealing with ATCA-related products.

The solution presented here fully complies with all the most recent revisions of specifications that are required for an ATCA board to properly operate in an ATCA shelf, communicate with the redundant Shelf Manager (ShM) and host Advanced Mezzanine Cards (AMCs). Full Electronic-Keying (EK) functionality is present on the LLRF CB supporting such protocols as PCI Express (PCIe), Gigabit Ethernet (GbE) and proprietary Low Latency Links (LLL) making it possible to route connections between all the boards in the system.

The IPMC solution presented here is mainly hardware independent as proper code organization allowed to separate low-level device drivers and high-level application logic dealing with the ATCA standard, which makes it portable to new carrier board designs.

Index Terms—**Advanced Telecommunications Computing Architecture, Intelligent Platform Management Interface, Intelligent Platform Management Controller, XFEL, Electronic Keying, High-Energy Physics, ATCA, IPMC**

I. Introduction

Modern high-energy physics (HEP) and other big-scale experiments and scientific undertakings require a great level of reliability, serviceability and availability as far as the control subsystems of these activities is concerned [1]. For many years this goal was not an easy thing to accomplish since pieces of equipment used in the experiments, often coming from various vendors, were not compatible with each other due to lack of a widely accepted standard which the production and application of this equipment could be based on. In mid-2004, however, a new standard emerged. Advanced Telecommunications Computing Architecture (ATCA), although designed primarily for telecom industry, has found its use in other applications, including high-energy physics experiments and accelerator control systems [2]–[4]. At the heart of the coveted reliability and availability features of ATCA stands the Intelligent Platform Management Interface (IPMI) specification [5] which has been adapted by ATCA as the communication protocol of the standard. Each ATCA board is equipped with an Intelligent Platform Management Controller (IPMC) which is responsible, among others, for controlling the hot-swap activation and deactivation of the board, managing the on-board sensors, verifying that the values given by them are limited to specific thresholds [6]. These modules monitor the most crucial components of the whole system and play an important role in early detection of any failures that might arise. This, in turn, enables the user to remove or replace a faulty module or take necessary actions to prevent the upcoming failure. Any messages that need to be delivered to the active Shelf Manager (ShM) concerning the aforementioned events are processed and sent by the IPMC. That is why it is a vital component of the system and its proper operation is crucial in order for the high levels of reliability and availability to be upheld [7].

Implementations of IPMC are hardware-dependent and thus ready-made solutions are neither easily obtainable nor relatively cheap. Although there exist reference designs offered by some vendors, their flexibility implies resource wastage when implemented in less-demanding applications. On the other hand, more robust projects which include specialized hardware require drivers for these devices to be written separately from the available IPMC solutions and integrated into the ready-made product, e.g. for Advanced Mezzanine Card (AMC) management [8] or Electronic Keying (EK) purposes [9], [10]. This could be a difficult task to accomplish taking into consideration limited access to the source code offered by some vendors.

Taking into consideration the fact that many projects include development of proprietary, but still standard compliant, hardware it is not an issue to create a proprietary IPMC as well. This article focuses on one such application developed for the Low Level RF Control System of the XFEL Accelerator designed in the Department of Microelectronics and Computer Science (DMCS) at the Technical University of Lodz (TUL), Poland and describes the IPMC designed for ATCA Carrier Boards (CB) used in the system [11].

D. Makowski and P. Prędki are with the Technical University of Łódź, Department of Microelectronics and Computer Science, 90-924 Łódź, Poland (e-mail: ppredki@dmcs.pl, dmakow@dmcs.p.lodz.pl).

II. ATCA Carrier Board for LLRF System

In order for the CB to be compatible with other ATCA boards as well as ShM cards controlling the shelf, it needs to comply with the standard as far as both hardware and software are concerned [12].

The Carrier Board communicates with the ShM over a redundant I^2C bus called IPMB-0 and with up to three AMCs over IPMB-L. Other peripheral devices are connected to the one remaining I^2C bus. These include voltage, temperature and current sensors, I/O port expanders, an external EEPROM memory chip, integrated clock generator, cross-switches and an integrated clock buffer, see Fig. 1 [9].

III. IPMC Requirements for LLRF ATCA Carrier Board

The proprietary IPMC proposed by the authors needs to fulfill specific requirements that follow from the general requirements of the LLRF system architecture as well as the hardware available on the CB itself. These features are:

- Compliance with the basic IPMI commands required by the ATCA standard,
- Compliance with the PICMG 3.0 extension commands,
- Carrier board management including hot-plug functionality; EK functionality for PCIe, GbE, LLL; Blue LED control; Hardware Address (HA) recognition,
- Management of three AMC modules including hot-plug functionality, EK functionality, power supply control,
- External sensor monitoring for stand-alone devices (e.g. MAX6626) and integrated devices (e.g. ATC210) including voltage, temperature and current sensors,
- Debugging and diagnostic functionality,
- Economical, easy to implement and low-area solution

The messaging between the IPMC on the CB and one on the ShM takes into account the requirements and structure as shown in the version 2.0 of the IPMI specification and the PICMG 3.0 Revision 3.0 document [13]. All the messages that are mandatory as defined by those documents have been implemented allowing the Carrier Board to operate in all standard-compliant shelves. This implementation includes the EK feature which allows dynamic configuration of links between the boards in one shelf [14] as well as between the AMC modules and other devices in the shelf.

There are four I^2C channels supported by the IPMC. Two of those are used for the redundant Intelligent Platform Management Bus (IPMB-0) over which the CB communicates with the ShM, one is used for the local IPMB bus (IPMB-L) to which the AMC modules are connected. A maximum of three single-width AMC modules can be controlled by the CB. Again, the communication between these components is fully in accordance with the documents mentioned before as well as the AMC.0 Revision 2.0 specification [8] which describes in detail the communication patterns between CBs and AMCs.

The HA translates directly to a I^2C address the board needs to use on the IPMB-0 bus. Its recognition is done on the CB with the aid of the I/O expanders which collect this information from the backplane, as detailed in the specification. Two more crucial ATCA-specific elements are the hot-swap plug and the Blue LED. The former is required for the board to be safely removed and replaced on-the-fly without the need to stop the whole system and the latter indicates the state of the board to the user which can perform specific actions depending on this information. The CB IPMC is able to detect the changes of the hot-swap plug as well as control the Blue LED as defined by the requirements.

Although all the vendors of ATCA components boast full compliance with the specification, experience shows that this is not always true. While working with AMC modules and ATCA boards provided by several manufacturers it has been observed that some of the IPMI messages sent over the appropriate bus do not correspond to the ones described in the specification. Whether a matter of wrong bit or byte ordering or sending hot-swap messages in an incorrect order, such discrepancies do occur and can influence the proper operation of the system. In order to eliminate such errors in the IPMC development it has been designed with support of a debugging feature that can work with the IPMI Analyzer, developed in parallel, or send plain-text messages over the serial interface directly to the user without the need of extra equipment.

The IPMC reads data from all the sensors present on the CB and is capable of sending event messages to the ShM whenever any of the thresholds specified for any of the sensors is crossed. These messages are then interpreted by the ShM and stored in the System Event Log (SEL) where the system manager can spot unusual behavior and take actions accordingly, such as increasing the speed of fans in the shelf for overtemperature alerts. This mechanism greatly contributes to the overall stability and reliability of the system.

IV. IPMC Hardware and Software

Taking into consideration the requirements set for the IPMC, suitable hardware and software solutions needed to be found. Although some of them force the designer to use specific components, such as those for power distribution, there is a wide range of possibilities to choose from as far as the sensors, the microprocessor for IPMC or the software implementation are concerned.

The IPMC code can be easily divided into several parts which correspond directly to the LLRF system requirements for this solution. The major software components, which will be described in more detail in the next sections, are:

- IPMI library including functions dealing with all the base IPMI and PICMG 3.0 commands,
- Communication section including functions dealing with messaging between MMC and IPMC and IPMC and ShMC,
- AMC management section for proper AMC activation, deactivation and EK operation,
- Sensor management section for initializing, controlling and event detection from on-board sensors,
- Firmware upgrade section as specified by the HPM.1 documentation for both the IPMC and other programmable devices,
- Debugging section for sending human-readable output over a serial interface for diagnostic and verification purposes

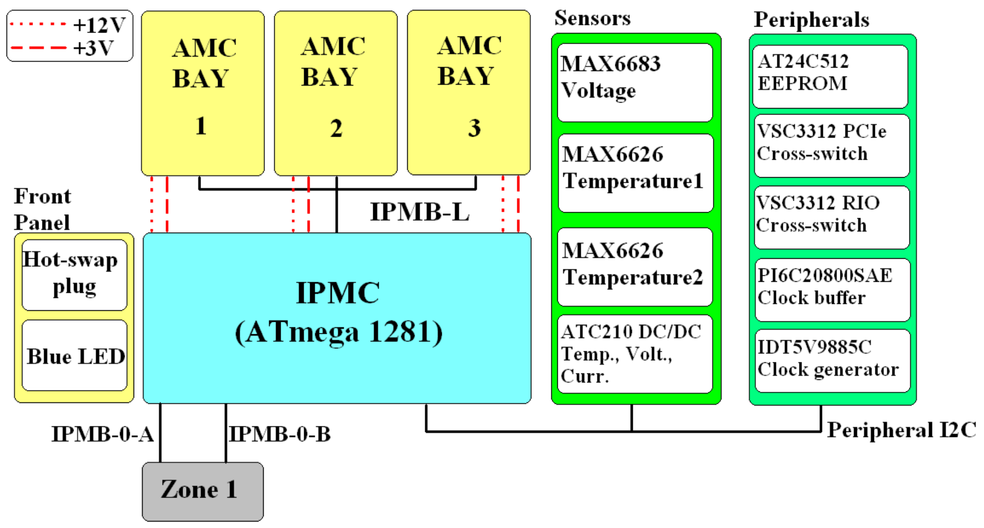

Fig. 1. IPMC and peripheral devices

A. Microprocessor Selection

The IPMC on the CB is implemented using an Atmel AVR ATmega 1281 microprocessor. It comes with 128 kB of Flash program memory, 8 kB of SRAM and 4 kB of internal EEPROM. This chip has been chosen in order to utilize its resources as optimally as possible. The program memory is filled in around 70% while the SRAM in just below 90%. The internal EEPROM is used for storing the debug messages sent over the serial interface. Also, what is more important, it contains the Field Repeatable Unit (FRU) information each IPMC is required to implement. Data stored there is supplied to the ShM and details all of the crucial features of the CB such as power consumption, number of supported AMCs or supported interfaces used in EK negotiations. Although this information may be modified at run-time it is a very rare occurrence and thus storing it in a non-volatile memory is justifiable. This is also a requirement of the standard.

This microprocessor was chosen as the best compromise between price and efficiency. The memory as well as peripheral resources usage shows that a smaller chip might not have been able to deal with the IPMC functionality and a bigger one would introduce resource wastage. Also, the tools used to write the source code are easily obtainable and free. These include the GNU toolchain for C-language adapted for the AVR architecture and an Integrated Development Environment available from the Atmel website. The chip itself is not expensive either compared with more robust microprocessors.

B. Code Organization

As mentioned before, the internal operation of the IPMC is hardware-dependent. It needs to know what kind of sensors and other peripheral devices are present on the board and how to communicate with them. Also, the hardware implementation of I^2C controllers may vary from board to board and the IPMC has to be able to read and send the IPMI messages over the IPMB buses. The ATmega 1281 present on the CB has only one built-in Two-Wire Interface (TWI) controller used for communication with AMC modules and all the other I^2C buses are managed by external chips connected to the IPMC by a parallel bus. Other implementations may use microprocessors with a higher number of built-in controllers or use other devices altogether such as Field Programmable Gate Arrays (FPGAs).

The IPMC source code, written in C language, is invulnerable to some of these variations because of the way it has been structured. Most of the interactions between IPMC and the low level devices have been wrapped in high level functions with an API available in form of doxygen generated documents. Thus, the device driver part of the design is separated from the logic of the program. Similarly, the reception and transmission of the IPMI messages is largely independent of the logic processing this information. The low level I^2C device driver monitors the bus, collects the data and then inserts it into a receive queue where it is read from by a high level function that analyzes the whole packet. Likewise, the response message is packed into a send queue and the low level device driver deals with sending it over the I^2C interface. If a change to hardware needs to be made only those drivers need to be changed while the core logic remains untouched.

For debugging purposes an additional application has been developed that gives the user control over most of the on-board peripherals. Presented with a graphical user interface he or she is able to read the values of the sensors or the I/O port expanders, reconfigure said sensors, check and change the LEDs states, etc. [15]. Such a tool is invaluable in the early stages of system development enabling to verify both the hardware and software components. The functions dealing with communication with this application can be switched on or off at compile time of the IPMC and, if on, they do not interfere with the IPMC functionality.

C. Real-Time Pseudokernel

It is understandable that a system controlling high-energy physics experiments of which high reliability and availability

TABLE I
MAIN IPMC LOOP

Action	Associated Function
Process CB-related events	processEvent()
Process AMC-related events	processAMCEvent()
Process ShM IPMI messages	dispatchMsg()
Process AMC IPMI messages	dispatchIPMB_L()

are required should present response times to various event which are as low as possible. The external events include arrival of IPMI messages, sensor interrupts or user hot-swap interaction. Fast message processing maximizes the utilization of the bus because no timeout message repetitions occur if the original message is handled and responded to quickly. Since a fully-featured real-time operating system would consume too many resources and be too complicated in implementation and analysis, a hybrid pseudokernel has been implemented to perform real-time actions for this application. The idea of event-driven cyclic executives in conjunction with external device interrupts has been employed [16]. The Interrupt Service Routines (ISR) have been reduced to minimum in order to maximize the response time of the application. Nested interrupts are disabled by default in ATmega1281 and are not enabled in the proposed solution in order to avoid the overhead associated with pushing and popping register values onto and from the stack. Thus, the faster one interrupt is serviced the sooner another one will be. Actually, the major role of ISRs in this solution is feeding the event-driven part of the code organization with specific events where the cyclic executives nature of the application takes over. This approach involves executing short processes in a continuous loop as depicted in TABLE I.

The CB-related events include opening or closing the hot-swap handle by the user, alerts from external sensors, requests for EK link reconfiguration, I^2C bus stuck alerts, watchdog alerts, requests for IPMI message transmission to ShM. The AMC-related events include requests for TWI reconfiguration, requests for IPMI message transmission to AMC, insertion/removal of AMC alert, requests for sending a ping message to AMC. This way, all of the major events that influence the operation of IPMC are dealt with in one place where it is easy to control and modify them. For example, adding or removing a sensor would only require to add an event to the event list and no changes to the core loop would have to be made.

D. IPMI Message Processing

The IPMI message processing is a crucial activity of the main loop of IPMC. The communication between the CB and the ShM or AMCs should be carried out as smoothly as possible and, although the IPMI standard assumes the possibility of resending unanswered requests, these occurrences should be limited to a minimum in order not to overload the buses. Taking into account the high level functions the whole process can be subdivided into four parts:

- Read an IPMI message from the receive buffer,
- Process the message,
- Formulate the response and put it in the transmit buffer,
- Indicate a message ready event.

Before the first action and after the last one the low level drivers are responsible for putting the received message into the input buffer and transmitting the response out of the output buffer, respectively. The input and output buffers are implemented in form of First-In First-Out (FIFO) queues, eight messages deep, where each message can hold up to 32 bytes, which is also defined by the standard.

The first function merely copies the message indicated by the read pointer of the FIFO queue to a local buffer and passes control to the processing function. This procedure verifies the checksum of the message and analyzes the frame itself according to the IPMI core specification as well as the PICMG 3.0 documentation. The messages are divided into several groups following the division of the latter document, see TABLE II. The first five categories represent the core IPMI specification commands while the last two follow the PICMG 3.0 commands extension.

TABLE II
IPMI COMMANDS CATEGORIES

Command Category	Associated File
IPM Device "Global" Commands	ipmi_global.c
BMC Watchdog Timer Commands	ipmi_watchdog.c
Event Commands	ipmi_events.c
Sensor Device Commands	ipmi_sensors.c
FRU Device Commands	ipmi_fru.c
AdvancedTCA	ipmi_picmg.c
AMC Communication Commands	ipmi_amc.c

If the checksum verification was successful the read pointer in the input FIFO queue is moved. During the processing of the message appropriate actions are taken depending on what the message was. These actions may be visible to the user (e.g. blinking the front panel LED) or hidden from the user (e.g. changing the threshold value of a certain sensor). Some messages are not followed by an action altogether - data may be simply gathered corresponding to the original request (e.g. FRU information, Sensor Data Repository (SDR) information). Regardless of the resulting activity, a response is formed with a suitable Completion Code (CC) and the overall message format as specified in the IPMI documentation.

Afterwards, the message is copied from the local buffer to the output FIFO queue. No transmission is made but an event is asserted indicating that the low level I^2C device driver should try sending the response the next time it is called from the main program loop.

The scenario described above holds true for cases when the originally received message was a request. For a received response message, no further response formulation is required and no event indication takes place, obviously. However, the output FIFO queue read pointer is updated following a successful response reception.

V. UPGRADE FUNCTIONALITY

A. HPM.1 IPMC Upgrade

The proposed IPMC solution implements the HPM.1 functionality which defines management firmware upgrade capa-

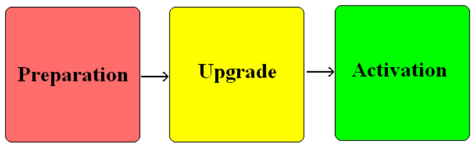

Fig. 2. HPM.1 Upgrade Stages

bility [17]. The main idea behind this specification is standardization of firmware upgrade procedures as far as management controllers, i.e. IPMC, Module Management Controller (MMC), are concerned. Before the introduction of this document in mid-2007 every vendor had used their own proprietary solution for firmware upgrade. Thus, for systems where components from different manufacturers where used, the problem of upgrading those various elements was significant. Now this issue is non-existent since a so-called upgrade agent following the specification is able to upgrade the firmware of every HPM.1-compliant device, including the LLRF carrier board IPMC.

The upgrade procedure uses IPMI-based messages and is capable of transmitting the new firmware over the same IPMB-0 or IPMB-L buses the rest of the communication takes place. However, there is also an option of sending the firmware directly over the payload interface. The developed solution supports only the former method.

The IPMC is capable of receiving the updated software without the need of terminating normal operation. The new firmware is stored in a buffer area in memory and the actual update may be deferred to a later, more convenient point in time. A rollback mechanism is also being developed in case the new firmware contains serious flaws which disable proper operation of the IPMC. The software upgrade process is divided into three stages as presented in Fig. 2. In the preparation step the upgrade agent verifies whether the upgrade image of the new firmware and the target device have the same manufacturer and device type and are altogether compatible. After that, the upgrade stage begins where the upgrade agents goes through upgrade action records, which contain the actual firmware data that should be stored on the target board. The data structure is of no interest to the upgrade agent so it is not important what kind of component is updated as long as the IPMC understands and correctly processes the incoming messages. The final stage is activation, which may happen instantaneously or be deferred. No self-tests are implemented in the current version of the IPMC.

B. *Upgrade of Programmable Devices Firmware*

The HPM.1 specification is not limited to upgrading management controller firmware. Any kind of device can be updated using the procedure described in the preceding subsection as long as the IPMC is able to process the upgrade action records received from the upgrade agent. The firmware of programmable devices available on the Carrier Board, like microcontrollers, FPGAs or DSPs can be uploaded in similar way as IPMC firmware. FPGA devices can be programmed using parallel bus or serial interface based on a JTAG standard. The JTAG standard allows to connect more than one device, therefore more FPGA devices or PROMs (Programmable Read Only Memories) can be included into JTAG chain. The JTAG interface of the main Xilinx FPGA present on the Carrier Board is connected to a multiplexer. The JTAG signals available on AMC modules' connectors are also supplied to the multiplexed. Therefore the devices in JTAG chain, selected by the IPMC, can be programmed using the same HPM.1 protocol that was implemented for IPMC microcontroller firmware upgrade. The programmer for FPGAs was implemented in IPMC according to Xilinx note [18]. The firmware of DSP processor is stored in non-volatile flash memory controlled by SystemACE controller [19]. The flash memory can be also programmed using JTAG.

VI. Tests and Evaluation

The ATCA-based LLRF Conrol System was tested in the FLASH free-electron laser at DESY, which is also used as a test platform for the systems which will be ultimately applied in the XFEL accelerator. Three test sessions were carried out in January, March and September 2009 [20]. The IPMC functionality of the LLRF CB was one of the test subjects as the device itself, equipped with AMC modules and a rear transition module (RTM), was used to control the superconducting cavities. The task of the IPMC was to properly activate the CB inside the ATCA shelf which is necessary for further activation of AMC modules. All three AMC slots were occupied and all three modules were controlled by the IPMC. The activation process was completed successfully and the PCI Express (PCIe) links [21] were established between them and the PCIe switch on the Carrier Board as well as an external PC station taking advantage of the EK functionality. These links are used for sending configuration parameters to the LLRF controller [9]. They can be transmitted over Ethernet connection to the PC and, subsequently, over PCIe to other devices [22].

More tests have been carried out in laboratory environment with emphasis put on proper AMC management. A Pigeon Point ShMM-500 ShM has been used for shelf management with the LLRF CB being the only ATCA board in the shelf during testing. The following AMC modules have been thoroughly tested:

- TEWS TAMC900 8 Channel 105 MSps 14 Bit AD Converter,
- Emerson PrAMC-6210 PowerPC based AdvancedMC Module,
- NAT NAMC-8560-xE1/T1/J1,
- SanBlaze SAS/SATA Hard Drive Module SB-AMC-HD.

For all the AMC modules full activation and deactivation was achieved. For the TEWS TAMC module, which supports PCIe, EK negotiations allowed it to be connected to the PCIe switch present on the LLRF CB proving that the EK functionality is properly implemented in the IPMC.

All the goals of the tests have been reached during the sessions and it has been shown that the ATCA-based solution can be used in situations where processing of many signals of small levels is required with as little noise as possible and with latencies of a few hundred nanoseconds.

VII. Summary

The proposed IPMC has done its job well as the heart of the CB used in the ATCA-based LLRF system for FLASH tests. Few problems have been encountered in the first sessions all of which have subsequently been eliminated. The IPMC has been shown to be able to work on CBs equipped with different sensors taking advantage of the driver / logic separation. The activation and deactivation procedures, defined in the PICMG 3.0 specification, have been followed directly as have been interactions with AMC modules and EK processes. Both PCIe and Gigabit Ethernet (GbE) links have been established.

Utilization of the HPM.1 firmware upgrade capability makes it easy for changing the IPMC even during operation of the system which is a much sought-after feature in accelerator applications and other HEP experiments. The same can be said about the hot-swap functionality that comes with the ATCA standard enabling removal and replacement of devices without disrupting the system. Thus, all the requirements imposed by the LLRF control system on the IPMC have been met.

By proving all the above functionality it has been shown that it is possible to develop and apply a proprietary IPMC code able to work in conditions requiring high availability and reliability rates proposed by the ATCA standard. Such a solution is economically sound when compared to those offered by major vendors dealing with this architecture at the same time giving the firmware developer full control of the features that need to be implemented for a given application which, in turn, assures that the utilization of all the system resources will be optimal.

Acknowledgment

The research leading to these results has received funding from the European Commission under the EuCARD FP7 Research Infrastructures grant agreement no. 227579 and Polish National Science Council Grant 642/N-TESLA-XFEL/09/2010/0. The authors are a scholarship holders of project entitled "Innovative education ..." supported by European Social Fund.

References

[1] R. Larsen, “Advances in developing next-generation electronics standards for physics,” in *Real Time Conference, 2009. RT ’09. 16th IEEE-NPSS*, May 2009, pp. 7 –15.

[2] X. Hao, W. Qiang, L. Lu, J. Dapeng, L. Zhen’an, J. Lang, S. Lange, L. Ming, and W. Kuehn, “Application of ATCA in trigger and DAQ system for experimental physics,” in *Real Time Conference, 2009. RT ’09. 16th IEEE-NPSS*, May 2009, pp. 571 –573.

[3] A. P. Lowell and W. Sun, “Real-time X-ray tomosynthesis imaging using an ATCA general-purpose data acquisition and analysis platform,” in *Nuclear Science Symposium Conference Record, 2008. NSS ’08. IEEE*, Oct. 2008, pp. 27 –31.

[4] B. Goncalves, J. Sousa, A. Batista, R. Pereira, M. Correia, A. Neto, B. Carvalho, H. Fernandes, and C. Varandas, “ATCA advanced control and data acquisition systems for fusion experiments,” in *Real Time Conference, 2009. RT ’09. 16th IEEE-NPSS*, May 2009, pp. 28 –34.

[5] IPMI v2.0 rev. 1.0 specification. [Online]. Available: http://www.intel.com/design/servers/ipmi/spec.htm/

[6] J. Lang, K. W. Ming Liu, Qiang Wang, and H. X. Zhen’an Liu, “Intelligent platform management controller for ATCA compute nodes,” in *Real Time Conference, 2009. RT ’09. 16th IEEE-NPSS*, May 2009.

[7] K. Przygoda, A. Piotrowski, G. Jablonski, D. Makowski, T. Pozniak, and A. Napieralski, “ATCA-based control system for compensation of superconducting cavities detuning using piezoelectric actuators,” in *Nuclear Science Symposium Conference Record, 2008. NSS ’08. IEEE*, Oct. 2008, pp. 38 –43.

[8] PICMG AMC.0 AdvancedMC Mezzanine Module R2.0. [Online]. Available: http://www.picmg.org/v2internal/specifications.htm/

[9] D. Makowski, W. Koprek, T. Jezynski, A. Piotrowski, G. Jablonski, W. Jalmuzna, and S. Simrock, “Interfaces and communication protocols in ATCA-based LLRF control systems,” *Nuclear Science, IEEE Transactions on*, vol. 56, no. 5, pp. 2814 –2820, Oct. 2009.

[10] V. Ricchiuti, A. Orlandi, and G. Antonini, “High speed serial links characterization for ATCA backplanes,” in *Signal Propagation on Interconnects. SPI 2008. 12th IEEE Workshop on*, May 2008, pp. 1 –4.

[11] S. Karstensen, I. Sheviakov, L. Frohlich, K. Rehlich, M. Staack, and P. Vetrov, “Machine protection system (mps) for the xfel,” in *Real Time Conference, 2009. RT ’09. 16th IEEE-NPSS*, May 2009, pp. 16 –21.

[12] W. Qiang, A. Jantsch, J. Dapeng, A. Kopp, W. Kuehn, J. Lang, S. Lange, L. Lu, L. Ming, L. Zhenan, L. Zhonghai, D. Muenchow, J. Roskoss, and X. Hao, “Hardware/software co-design of an ATCA-based computation platform for data acquisition and triggering,” in *Real Time Conference, 2009. RT ’09. 16th IEEE-NPSS*, May 2009, pp. 485 –489.

[13] PICMG 3.0 advancedTCA Base R3.0. [Online]. Available: http://www.picmg.org/v2internal/specifications.htm/

[14] P. Predki and D. Makowski, “Hot-plug based activation and deactivation of ATCA FRU devices,” in *Mixed Design of Integrated Circuits Systems, 2009. MIXDES ’09. MIXDES-16th International Conference*, June 2009, pp. 119 –122.

[15] J. Wychowaniak, P. Predki, D. Makowski, and A. Napieralski, “Diagnostic application for development of custom ATCA carrier board for LLRF,” in *Mixed Design of Integrated Circuits Systems, 2009. MIXDES ’09. MIXDES-16th International Conference*, June 2009, pp. 97 –102.

[16] P. A. Laplante, *Real-Time System Design and Analysis*. Wiley, 2004.

[17] PICMG hpm.1 Management Firmware Upgrade Capability. [Online]. Available: http://www.picmg.org/v2internal/specifications.htm/

[18] Xilinx, “Xilinx Xilinx In-System Programming Using an Embedded Microcontroller,” in *XAPP058*, March 2009.

[19] ——, “System ACE CompactFlash Solution,” in *DS080*, October 2008.

[20] S. Simrock, “Demonstration of ATCA based LLRF control system at FLASH,” in *ICALEPS 2009*, October 2009.

[21] T. Kucharski, A. Piotrowski, D. Makowski, and G. Jablonski, “PCIExpress communication layer for ATCA-based linear accelerator control system,” in *Mixed Design of Integrated Circuits Systems, 2009. MIXDES ’09. MIXDES-16th International Conference*, June 2009, pp. 140 –144.

[22] S. Simrock, L. Butkowski, M. Grecki, T. Jezynski, W. Koprek, G. Jablonski, W. Jalmuzna, D. Makowski, A. Piotrowski, and K. Czuba, “Evaluation of an ATCA based LLRF system at FLASH,” in *Mixed Design of Integrated Circuits Systems, 2009. MIXDES ’09. MIXDES-16th International Conference*, June 2009, pp. 111 –114.