Firmware Upgrade in xTCA Systems

D. Makowski, *Member, IEEE*, G. Jabłoński, P. Perek, A. Mielczarek, P. Prędko, T. Jeżyński, H. Schlarb, A. Napieralski, *Senior Member, IEEE*

Abstract—The Advanced Telecommunications Computing Architecture (ATCA) and Micro Telecommunications Computing Architecture (μ TCA) standards, collectively known as xTCA, provide a flexible and scalable infrastructure for designing complex control and data acquisition systems. The xTCA standards are becoming more and more popular in physics applications. Programmable devices, such as Field Programmable Gate Arrays (FPGAs), conventional and Digital Signal Processors (DSPs) are present on Advanced Mezzanine Card (AMC) modules and ATCA blades used in the xTCA crates. Those devices typically boot from non-volatile memories available on the modules.

This paper deals with an universal framework and set of tools for upgrading firmware at such devices in xTCA systems. The proposed framework uses a fat pipe region interface of μ TCA backplane for firmware data transmission and the Intelligent Platform Management Interface (IPMI) standard for PROM memory management and control of the upgrade procedure. This is the world's first attempt to implement the firmware upgrade in μ TCA system not using JTAG Switch Module (JSM).

Index Terms—Programmable Devices, Field Programmable Gate Array, Firmware Upgrade, Advanced Telecommunications Computing Architecture, Micro Telecommunications Computing Architecture, High-Energy Physics

I. INTRODUCTION

THE modern control systems used in High Energy Physics (HEP) are designed with programmable devices, like processors, Digital Signal Processors (DSPs) or Field Programmable Gate Arrays (FPGAs). The firmware for those system components is usually stored in non-volatile memories. During firmware development dedicated programmers are used. The tools provided by manufactures allow to download firmware to non-volatile PROMs (Programmable Read-Only Memories) and provide useful debugging functionality. The programmers can be connected using standard JTAG or proprietary debug connectors. Since xTCA systems are composed of various Advanced Mezzanine Cards (AMCs) or Advanced Telecommunications Computing Architecture (ATCA) modules, which may include various programmable devices, a large number of different tools can be required. The μ TCA or ATCA chassis may house, respectively, up to 12 or 16 cards. In case of laboratory development, the firmware upgrade is performed using only a few dedicated programmers, upgrading each programmable device one by one. The situation is more difficult when the xTCA hardware is used to control a complex machine. In this case, the devices

cannot be easily accessed. For example, the Low Level Radio Frequency (LLRF) control system of the European X-ray Free Electron Laser (EuXFEL) will be installed inside an accelerator tunnel. The system will be composed of hundreds of programmable devices. Therefore, the firmware upgrade cannot be performed using only programmers. Along with the deployment of complex control or data acquisition systems, the need for remote and automated firmware upgrade solution becomes more and more important. An universal framework and set of tools for upgrading firmware in xTCA systems has been developed and tested. The proposed framework uses a fat pipe region interface of μ TCA backplane for firmware data transmission and the Intelligent Platform Management Interface (IPMI) standard for PROM memory management and control of the upgrade procedure. The proposed Firmware Upgrade Framework (FUF) has been tested with the μ TCA-based LLRF control system of the Free Electron Laser at Hamburg (FLASH).

II. FIRMWARE UPGRADE IN DATA ACQUISITION AND CONTROL SYSTEMS

The xTCA standards do not specify the firmware upgrade procedure for payload devices [1]–[4]. The firmware upgrade can be performed using different methods during debugging, work in laboratory or in-system development.

A. xTCA Standards Family

The μ TCA and ATCA standards, collectively known as xTCA, provide a flexible and scalable infrastructure for designing complex control or data acquisition systems. Both were designed to provide developers with methods for achieving high Reliability, Accessibility and Serviceability (RAS) in high-throughput computing and telecommunications platforms [4], [5].

The ATCA chassis contains power entry modules, a fan tray, Shelf Managers, a backplane and a number of card slots. Almost every built-in system is doubled to ensure high reliability by means of redundancy. Another factor increasing the reliability is continuous monitoring.

The Micro Telecommunications Computing Architecture (MTCA, μ TCA, μ TCA) may be considered as a smaller and cheaper variant of the ATCA [2]. μ TCA chassis hosts up to 12 Advanced Mezzanine Cards (AMCs) [6]. The AMC/ μ TCA standards offers similar thermal, power and link management capabilities as ATCA.

B. Programmable Devices in xTCA Systems

Programmable devices provide flexible resources that can be easily adapted to increasing demands of modern control

Manuscript received June 10, 2012; revised February xx, 2012.

D. Makowski, G. Jabłoński, P. Perek, P. Prędko, A. Mielczarek, A. Napieralski are with the Technical University of Łódź, Department of Microelectronics and Computer Science, 90-924 Łódź, Poland (e-mail: dmakow@dmcs.p.lodz.pl).

T. Jeżyński, H. Schlarb are with the Deutsches Elektronen-Synchrotron (DESY).

TABLE I
SUMMARY OF MEMORY SIZES IN CURRENTLY AVAILABLE
PROGRAMMABLE DEVICES

Device type	Memory size
DSPs	a few kB – a few MB
Microcontrollers	a few kB – tens of MB
FPGAs	up to 7.8 MB (Virtex 5) up to 53.3 MB (Virtex 7)

and data acquisition systems. Because of high complexity and wide range of functionality, the systems include various types of programmable devices. Depending on the needs, microcontrollers, DSP processors and FPGA devices are applied. The most important feature from the firmware upgrade procedure point of view of those devices is the size of program memory. It explicitly influences the amount of data which has to be sent to the device and, in consequence, the programming time.

Table I presents program memory sizes for programmable devices which are currently available on the market. In case of the microcontrollers and DSP processors the size of the program memory usually equals kilobytes, up to a few megabytes, taking into consideration the solutions without operating systems. FPGAs are more demanding. In their case, sizes of memory for programming range from several to tens of megabytes [7], [8]

1) *Firmware Upgrade in Laboratory*: Due to the size and write speed of currently available Flash memories, their programming time is in range of few seconds to several minutes. When the system is in the initial development stage, the programs or bitstreams (FPGA configuration files) are loaded directly to the SRAM memory of the device using dedicated programmers. Moreover, these devices often provide essential trace and debug capabilities. In such an environment the remote upgrade of Flash memories does not increase the comfort nor reduce the time of programming the device. Nevertheless, it is an optimal phase to evaluate the firmware upgrade feature, as the consequences of improper operation are negligible and the debugging is relatively easy.

2) *In-System Firmware Upgrade*: When the system is finally assembled and e.g. mounted in a dedicated shelf, it is no longer easily accessible. Use of debugging tools is usually limited or even impossible at all, which is the case with systems installed inside the accelerator tunnel. If the system is not proven to be free of bugs some mechanism for upgrading the firmware has to be implemented. On the other hand when the system is known to work perfectly, it is still worth to implement this functionality to have the possibility of extending the set of its features.

C. Firmware Upgrade Methods

1) *JTAG Interface*: The programming of programmable devices using JTAG is described by the IEEE 1532-2002 standard [9]. When the system is composed of several boards, making a large chain of all programmable devices is not practical. For VME systems the Module Test and Maintenance Bus [10] is a standard way of accessing JTAG chains of individual boards even today [11].

The μ TCA specification [2] defines the JTAG connections on the AMC modules and an appropriate backplane, that can be optionally connected to a JTAG Switch Module (JSM), providing independent access to the chains of all the boards in the system. Systems equipped with this module are available commercially [12]–[14], however the crates chosen for the EuXFEL control [15] do not support this type of connections.

2) *Direct Bitstream Upload*: If the FPGA firmware adheres to specific conventions, it is possible to configure the device via PCI Express, using Altera's Configuration via Protocol [16], [17] or partial reconfiguration [18], [19]. Using this method allows the firmware to be seamlessly updated, without the need to reenumerate the PCIe bus. However, the PCIe configuration (e.g. number and size of BARs) has to be fixed and known in advance.

3) *Indirect Programming*: In this programming method, the boot memory is configured indirectly via the device it configures, and then the device is programmed from this memory. This approach requires re-enumeration of the PCIe bus, as the programming process breaks the communication with the device. To ensure proper recovery when the system is inaccessible, it is important to have the possibility of restoring the programming capability if the upgrade process should fail. To obtain this functionality use of at least two Flash memories is recommended. One of them always contains the firmware a with working programmer module. The other may contain an user application with or without the programmer module. Another possibility is to use multiple revisions of the same Flash, although this approach is less reliable. The programmer can be also integrated with user firmware. If the Platform Flash is to be programmed by means of PCIe, the XSVP player [20] can be used, that only requires access to four pins of the FPGA, or the solution described in [21], requiring a slightly more complex module inside FPGA. If this indirect method is used via Ethernet, typically an embedded processor, such as Microblaze [22], is used.

Using multiple memories or multiple revisions (one for the programmer, optionally integrated with the known working revision of the firmware) requires some means of choosing the memory to boot from. Preferably, this task is assigned to the IPM controller (MMC in case of μ TCA or IPMC on ATCA-based systems).

4) *Hardware Platform Management IPM Controller Firmware Upgrade*: For xTCA systems, the on-board CPU and other hardware is known as payload. PICMG does not define the standard to configure the payload, leaving it to the system developer. It only defines the standard to upgrade the firmware of the IPM controller (MMC or IPMC), required on every board [23]. It can be used for our purposes if we treat the programmable device firmware (FPGA or DSP) as part of IPMC. The main drawback of this solution is the low throughput of the I²C bus (100 kbit/s) and therefore the long programming time of the payload device or memory.

III. LLRF CONTROL SYSTEM

The Low Level Radio Frequency (LLRF) system for Free-electron LAsER in Hamburg (FLASH) controls the phase and

amplitude of electromagnetic field inside superconducting cavities of a linear accelerator. Field stability of the order of 0.01% for amplitude and up to 0.01° in phase at 1.3 GHz is needed to provide required possible electron beam quality [24]–[27]. The control loop latency cannot be larger than a few μ s.

The system first measures field intensity in up to 32 accelerating cavities powered by one klystron (high power RF amplifier) and then calculates the corresponding phases and amplitudes. These parameters are sent to a control board, where all the vectors are added and compared with a set point. The controller calculates required field correction and actuates the vector modulator supplying the klystron with an RF signal.

The control over high-speed ADCs and DACs, data transfers over backplane or optical links and calculation of the vector sum is implemented using FPGAs. These operations may be efficiently realized in parallel hardware blocks. The control algorithms may be implemented using an additional DSP chip, that can execute complex floating point operations, or also using the FPGA resources. New FPGA devices offer slices containing wide multipliers and accumulators that are well suited for DSP operations.

The prototype LLRF control system of the FLASH accelerator is based on the μ TCA standard. The system uses all components required by the standard, such as CPU (Central Processing Unit), MCH (MicroTCA Carrier Hub) and dedicated AMC modules containing FPGA and DSP devices. Those modules include a signal processing module - the main controller (μ TC), digitizer cards (SIS8300), Vector Modulator (VM) and a timing card. The μ TCA chassis housing AMCs with available interfaces is presented in Fig. 1.

Fig. 1. LLRF submodules installed in μ TCA shelf

The μ TC board, equipped with Xilinx Virtex 5 FPGA and TigerSHARC DSP programmable devices, is intended to cooperate with a vector modulator RTM [28], [29]. Therefore, the JTAG chain of the μ TC includes the main FPGA and the devices available on the RTM: Spartan 6 FPGA and Platform Flash, see Fig. 2.

The board is equipped with three non-volatile SPI PROMs. The size of a single PROM is equal to 128 Mbits. The MMC

Fig. 2. Block diagram presenting programmable devices on μ TC board

(Module Management Controller) is used to select the active PROM using custom commands. The reconfiguration of the devices may be forced by Payload Reset IPMI command. The completion status of this operation can be monitored via MMC. The FPGA may access the SPI memories at run time, for both reading and writing. This enables storing of the DSP firmware in the Flash memory. As there are three SPI memories, one of them can store the emergency bootloader bitstream that can be used to program the remaining memories. The other two can store firmware for the FPGA and DSP. The RTM FPGA and Platform Flash can be programmed using the main FPGA on the μ TC as a JTAG programmer. All devices can be programmed using an additional JTAG connector and external programmer.

The VM RTM is dedicated to operate with μ TC AMC card. Both cards are connected via zone 3 connector that provides data, management and JTAG signals. The VM uses a Spartan 6 FPGA with a single Platform Flash. The JTAG chain is connected to the main FPGA of μ TC and included in its chain when the RTM module is plugged into the μ TCA chassis, see Fig. 2. There is no control of Platform Flash firmware revision so only a single revision is provided. The FPGA is rebooted automatically after successful upgrade of PROM.

The SIS8300 board has only a single Platform Flash PROM (32 Mbits) [30]. The PROM can be programmed only via JTAG. For that purpose, the entire JTAG chain, consisting of the Platform Flash and FPGA can be connected to the JTAG connector, on-board MMC or the FPGA pins as presented in Fig. 3. Switching between MMC and FPGA programmer is performed by the MMC controller via custom IPMI commands. During the normal operation the FPGA can be used as a programmer to update the firmware stored in PROM revisions.

Fig. 3. Block diagram presenting programmable devices on SIS8300 board

The Platform Flash is capable of storing two firmware revisions. It is possible to switch between them via the MMC controller. The first revision should contain the emergency bootloader, the second one can be updated independently and should contain the actual user firmware. It is possible to force reboot of the FPGA by the MMC controller, but it is not possible to read the boot completion status.

The DAMC 2 board has a single Platform Flash PROM connected in serial mode (32 Mbits) [31]. The JTAG chain includes the PROM and the main FPGA. In addition the FMC (FPGA Mezzanine Card) module can be included in the chain. The JTAG chain can be programmed using JTAG connector, JTAG pins available on the AMC connector, MMC and the main FPGA (configuration is available via custom IPMI commands). The detailed description of JTAG configuration and programming possibility can be found in datasheet [31]. The Platform Flash is capable of storing two firmware revisions as in case of SIS8300 card. It is possible to switch between them using the MMC controller.

A. Requirements for LLRF Firmware Upgrade

The firmware upgrade infrastructure should allow to download firmware for programmable devices of the following LLRF system components installed in the μ TCA shelf:

- Digital signal processing controller module (μ TC) [29],
- Vector Modulator (μ VM) [28],
- SIS8300 digitizer module (μ ADC) [30],
- General purpose FMC-carrier module (DAMC2) [29], [32].

The framework is supposed to be used for in-system firmware upgrade, therefore the programming time is not so important. Two types of non-volatile PROMs are used in the LLRF system: SPI and Platform Flash. Therefore, the infrastructure should allow to reprogram the PROMs, change the active revision (rollback after unsuccessful upgrade) and reboot the FPGA. The HPM.1 standard can be applied to firmware upgrade. However, the IPMI protocol of μ TCA uses the I²C protocol for data transmission. Therefore transmission of a few megabits firmware takes a significant amount of time (e.g. 15 minutes for a single revision of Xilinx XC5VLX50T). The PCIe protocol can be used to make the transfer faster [33].

IV. FIRMWARE UPGRADE INFRASTRUCTURE

Upgrading the firmware of programmable devices in xTCA-based systems requires cooperation of several various subsystems. Figure 4 presents an infrastructure designed for firmware upgrade in μ TCA-based LLRF control system. It includes components from various layers of the system, from hardware (FPGA) and management (MMC) to software (firmware upgrade agent). The FPGA firmware implements a PCIe module for data transmission and a programmer module which is responsible for programming Flash memory. Since various modules of LLRF systems use different PROMs, a XSVF player [20] has been used for Platform Flash and a custom programmer for SPI PROMs. The memory for programming or booting is selected by the MMC, which can also initiate and monitor the booting process of the FPGA. Both mentioned

components are supervised by the Firmware Upgrade Agent - application running on the CPU module ensuring proper course of firmware upgrade procedure. The agent allows to program SPI PROMs and play the .xsvf files prepared for JTAG programming.

A. Firmware Upgrade Procedure

The firmware upgrade procedure consists of similar steps for all the supported types of boards (see Fig. 5). However, due to different types of memories used on particular boards the first part of the procedure - file preparation - is slightly different. In case of the SIS8300 and DAMC2 boards, which use Platform Flash PROM memory, the firmware upgrade flow starts with the creation of the bitstream file. Then, it is converted to an mcs file with Xilinx iMPACT tools. Then, the .xsvf file containing the JTAG operations is generated. The resulting file is utilized by the Firmware Upgrade Agent for reprogramming of the Platform Flash. Then, the appropriate revision of the bitstream is selected, the FPGA boot is forced and finally the PCIe hot-swap mechanism must be invoked so the operating system can recognize the board in PCIe space.

In case of the μ TC board, which is equipped with SPI Flash memories, the first step is also creation of the bitstream. Due to the fact that it can be directly used for memory programming, the next step is selection of the memory bank. Then, the PROM is programmed and the FPGA is rebooted which causes the newly programmed bitstream to be loaded. Finally, the PCIe hot-swap must be performed.

Fig. 5. Firmware upgrade flow for SIS8300, DAMC2 boards (left) and for μ TC (right)

The μ TC is also equipped with an Analog Devices Tiger-Sharc DSP (ADSP TS-201) [34]. There are several ways to boot this device, such as booting from an external EEPROM chip, host booting and link booting. The latter approach is used in the case of μ TC. ADSP TS-201 is equipped with four high-speed serial links which can be used both for booting and for data transmission [35]. First, the DSP firmware needs to

Fig. 4. Block diagram of firmware upgrade infrastructure

be linked with an appropriate bootloader to form a file which can be transmitted via the link. The VisualDSP++ integrated development environment provides the options to do so. When configured for link booting, after reset, the DSP is waiting for 256 words of code to be received on any of the links. This kernel is then used to receive the rest of the firmware which ultimately overwrites the bootloader memory. The source of the data is the Virtex-5 FPGA, which reads the firmware from an external SPI-PROM chip. The FPGA is able to reset the DSP and force new firmware into it on demand.

B. Firmware Upgrade Agent

A key element of the system is the Firmware Upgrade Agent running on the CPU module. It implements algorithms ensuring execution of the firmware upgrade procedure. The procedure depends on the board type which is one of the input parameters of the application. The software supports programming SPI Flash (μ TC) and Platform Flash (SIS8300 and DAMC2) memories which are used to configure the FPGAs. The SPI FLASH memories are programmed directly using bitstream files. However, for programming Platform Flash memories the application uses xsvf files prepared for JTAG programming. For this reason the conversion of bitstream file to xsvf file is necessary. The firmware upgrade agent also gives the possibility to reboot FPGA from selected memory. Moreover, the software supports programming of the uVM board and DSP processor on the μ TC board.

The application provides a simple user interface which facilitates programming. Besides the type of the board, it requires information about the device type (FPGA or DSP), action which will be performed (memory programming or switching) and firmware file, if needed. Based on this information it carries out a specific firmware upgrade procedure. The program verifies input parameters and checks if the MCH is available. If the destination board is detected on the PCIe bus, the programming process starts. If not, a suitable message is displayed and the process is interrupted.

C. Support from MMC (IPMI)

Firmware upgrade requires support from the MMC of the programmed board. The MMC is responsible for selecting memory for FPGA booting, restarting FPGA and JTAG chain

TABLE II
IPMI CUSTOM COMMANDS

Board	Function	IPMI Command Code
SIS8300	Select active revision of memory	0x03
DAMC2	Select active revision of memory	0x01
	Configure JTAG chain	0x03
μ TC	Get PROM number	0x00
	Set PROM number	0x01
	Get FPGA activation status	0x05

configuration. The FPGA rebooting can be realized using Cold Reset or Payload Reset defined in PICMG IPMI and AMC specifications. However, for memory selection and JTAG chain configuration custom commands have to be implemented. Moreover, it is highly recommended that the MMC provides custom-defined commands for reading of selected memory number as well as for checking the FPGA status (whether it is booted or not).

The commands provided by boards, supported by the firmware upgrade infrastructure are summarized in Table II. In case of the SIS8300 board, there is only one command that allows the active revision of memory to be changed. It has an IPMI command code 0x03 and allows to send two bits that control firmware revision. In case of the DAMC2 board, there are two commands that allow the active revision of memory to be changed and JTAG chain to be configured. They have IPMI command codes 0x01 and 0x03, respectively. In case of the μ TC board, there are three custom commands available for firmware upgrade management. The first one of them allows the memory for FPGA booting to be selected. The second one can be used for checking the number of currently selected memory. And the last one allows the status of FPGA booting to be checked. They have IPMI command codes 0x00, 0x01 and 0x05, respectively.

All commands listed above are vendor specific commands and they belong to Controller-specific OEM/Group defined in IPMI standard. For this reason their Network Function code is 0x30.

The Firmware Upgrade Agent communicates with the MMC using functions selected from the ipmitool utility which supports communication with IPMI-compliant devices.

D. Data transmission

The data transmission interface used in the presented firmware upgrade infrastructure is PCI Express [33]. The firmware for FPGA devices, besides the application part, also includes a PCIe and a programmer cores. The PCIe core is responsible for receiving data from the firmware upgrade agent and its transmission to the memory programming module. The task of this module is programming the memory using SPI or JTAG interface, depending on the board type and installed memory.

The important element in the firmware upgrade procedure is PCIe hot-plug mechanism. It enables the PCIe bus to be refreshed after FPGA rebooting without the need of the CPU module to be restarted. There are two methods that can be used for the refreshing PCIe bus: hardware and software one. The first method requires support from the PCIe switch. When the board is deactivated or activated, a signal is sent to the PCIe switch and forwarded to a Root Complex (RC). The RC can refresh the bus when a hot-plug event is received. The software method uses a fakephp kernel module that allows it to emulate the extraction and insertion events. Both hardware and software PCIe hot-plug have been tested and finally the hardware one has been applied in the firmware upgrade procedure.

The situation is different in case of control signals transmitted from the user application to the MMC of the board e.g. in case of memory switching or FPGA rebooting. This information is sent in two steps. First, the application sends data to the MCH module using IPMI over Ethernet protocol. Then, the MCH transforms the received data and sends an appropriate command to the MMC via IPMI bus based on the I²C standard.

E. Automation

The process of memory programming has been automated with scripts which greatly facilitate the firmware upgrade process. The scripts are mainly responsible for the first step of firmware upgrade procedure – preparing input files for the firmware upgrade agent (see Fig. 5). In case of the μ TC board, it is limited to verifying if the file provided by a user is a correct bitstream file. However, in case of the SIS8300 and DAMC2 boards the scripts realize the conversion from .bit file to the .xsvf file as described in chapter IV-A.

After preparing the correct file with the firmware the scripts start the firmware upgrade agent that performs the remaining steps of firmware upgrade.

V. EXPERIMENTAL RESULTS

The developed firmware upgrade infrastructure has been tested in the digital laboratories at the Technical University of Lodz (TUL) and Deutsches Elektronen-Synchrotron (DESY). The tools were installed on a network drive, therefore the firmware upgrade tools have been available for various experiments and have been tested with the FLASH Cryomodule Test Bench Facility (CMTB).

TABLE III
SUMMARY OF TIME REQUIRED FOR PROM UPGRADE FOR AMC AND RTM BOARDS OF LLRF SYSTEM

Operation	μ TC	VM	SIS8300	DAMC2
Generation of XSVF	—	35 s	40 s	40 s
PROM switching	5 s	5 s	5 s	5 s
PROM erasing	60 s	30 s	30 s	30 s
PROM prog. and ver.	24 + 16 s	250 s	300 s	300 s
Total update time	105 s	320 s	375 s	375 s

A. Tests in μ TCA Shelf

The 12-slot μ TCA chassis from Elma, Adlink AMC-1000 CPU and N.A.T. MCH have been used during tests [15], [36], [37]. All AMCs and RTM, described in section III-A, have been tested. The LLRF AMCs and RTM uses Xilinx Virtex 5, Spartan 6 family FPGAs and TigerSharc DSP. The Xilinx ISE framework has been used to generate bit files for FPGA devices and the VisualDSP++ environment provided the DSP binary file.

1) *FPGA Booting Speed Test*: Since FUF assumes that the FPGA is loaded within 1 second, the default frequency of SPI memory used during booting of FPGA should be changed to at least 36 MHz to decrease the booting time. During the tests a 46 MHz clock has been applied (FPGA booting time 700 ms).

The total programming time of a single SPI PROM for the μ TC board is 105 seconds. This time includes switching to programmer, memory erasing, programming and verification. The detailed informations concerning those steps are summarized in table III.

The total programming time of boards equipped with Platform Flash was much longer than for μ TC and was equal to 320 s for VM and 375 s in case of SIS8300 and DAMC2 boards, see table III. The generated XSVF file should include commands for PROM verification after programming and FPGA reloading.

2) *DSP Booting Speed Test*: The ADSP TS-201 links consist of four differential pairs of data signals and one differential clock. Data is sent on every clock edge (double data rate). The transmission clock generated by the FPGA is 100 MHz, which translates to 800 Mbps of transmission speed. During booting, only raw data is transmitted with no protocol overhead. However, the FPGA state machine responsible for bootlader transmission introduces data fetching states which can be considered as 'empty states' in terms of bandwidth utilization. This resulted in a measured throughput of 630 Mbps. For tests in the μ TCA shelf a firmware file of 35 kB has been used with booting time of around 54 μ s. These results proved satisfactory although it should be possible to optimize the FPGA state machine in order to come closer to 100 % bandwidth utilization.

3) *Problems with PCIe Hot-Plug*: The Linux PCIe hot-plug implementation imposes some restrictions on the upgrade process. When the sizes of areas described by the Base Address Registers (BARs) are larger than the areas previously allocated for a given device, or the device has not been detected at boot-time, the address space for an upgraded device is not correctly allocated, making it impossible to use the device.

```

[ 335.930326] pci 0000:05:00.0: BAR 1: can't allocate mem resource [0xf8000000-0xf80fffff]
[ 335.930432] LLRFUTC: [llrfdrv_probe] DETECTED PCIE DEVICE
[ 335.930447] llrfutc 0000:05:00.0: enabling device (0400 -> 0402)
[ 335.930472] llrfutc 0000:05:00.0: setting latency timer to 64
[ 335.930478] LLRFUTC: [llrfdrv_probe] DEV INFO:
[ 335.930481] LLRFUTC: [llrfdrv_probe]         DEVFN 0x00 BUS_NUM 0x05 DEV_NUM 0x00, FUNC_NUM 0x00, BUS_FUNC 0x500
[ 335.930496] LLRFUTC: [llrfdrv_probe]         SLOT NR: 5
[ 335.930499] LLRFUTC: [llrfdrv_probe]         DMA MASK: 64 BIT
[ 335.930512] LLRFUTC: [llrfdrv_probe]         VENDOR_ID: 10EE DEV_ID: 0038
[ 335.930515] LLRFUTC: [llrfdrv_probe]         SUBVENDOR_ID: 10EE SUBDEV_ID: 0038
[ 335.930518] LLRFUTC: [llrfdrv_probe]         DEV FILE NAME: llrfutcs5
[ 335.931658] LLRFUTC: [llrfdrv_probe]         BAR0: START: 0xF4000000 END: 0xF7FFFFFF SIZE: 67108863 FLAGS: 0x20200
[ 335.931661] LLRFUTC: [llrfdrv_probe]         BAR1 NOT AVAILABLE
[ 335.931672] LLRFUTC: [llrfdrv_probe]         BAR2: START: 0xF3E00000 END: 0xF3E0FFFF SIZE: 65535 FLAGS: 0x20200
[ 335.931676] LLRFUTC: [llrfdrv_probe]         BAR3 NOT AVAILABLE
[ 335.931679] LLRFUTC: [llrfdrv_probe]         BAR4 NOT AVAILABLE

```

Fig. 6. Error message when allocating the BARs for an upgraded device is impossible

Fig. 6 presents an example error message in this case. This happens, because the memory allocation for the entire PCIe tree under Linux is performed at boot-time and no reserves for run-time modifications are made. The simplest solution to make the card usable is to reboot the CPU. One can also try to re-enumerate the entire PCIe device tree using *fakephp* driver, but this will enumerate also other devices connected to PCIe bus.

One of the possible solutions is to standardize the endpoint sizes (e.g. fix them at 64M/64M/64k).

VI. CONCLUSIONS AND PLANS FOR FUTURE

A. Improvements of Firmware Upgrade

The firmware upgrade system has not been designed for frequent, interactive upgrades during development, so the upgrade times have not been optimized. It is particularly noticeable in case of devices booted from Platform Flash, programmed by JTAG, when the simplest approach based on [20] has been applied. If necessary, the upgrade time can be shortened by means of application of one of the following solution:

1) *Parallel transfer of bitstream data to the FPGA*: Xilinx in [21] presents a module allowing to program Platform Flash PROMs with the reported speed of 9 s per 8 Mbit. The data to be programmed into the Flash memory is provided to the module using an 8-bit bus, simplifying the transfer of data via the PCIe bus. There are still some problems to be investigated with this solution:

- The programming clock may have to be reduced, depending on the quality of interconnection to the Flash memory, causing the reduction in programming speed.
- The module does not support advanced revisioning features, making it harder to recover after an unsuccessful firmware update.
- The Flash PROM should be directly connected to FPGA (only single device in JTAG chain).

Since the JTAG chains of SIS8300, DAMC2, uVM modules include more devices, this solution cannot be directly applied to this board and requires modifications in hardware.

2) *Application of Partial Reconfiguration*: The Dynamic Partial Reconfiguration (DPR) can be applied to reprogram

FPGA on-the-fly via the PCIe bus without the need for hot-plug nor re-enumeration of the PCIe tree. The Xilinx PlanAhead software allows to define boundaries between the fragments of the FPGA fabric that can be configured independently without interrupting the operation of the remaining parts. Xilinx provides the *ICAP_VIRTEX5* and *ICAP_VIRTEX6* primitives, that can be used to reprogram the FPGA from within. Thus, the external EPROM configuring the FPGA after the system boot can contain the configuration only for the PCIe endpoint block and the programmer. The rest of the FPGA can be configured from the CPU. According to [38], the configuration speed achieved for Virtex5 amounts to 1.6 MB/s.

This approach imposes some restriction on the firmware. The configuration of PCIe block (including the BAR sizes) has to be predefined, the area available for user logic is slightly reduced (max 80 % packing density achievable) and 10 % degradation in clock frequency is to be expected.

VII. CONCLUSIONS

The proposed firmware upgrade framework is the world's first implementation allowing the firmware of FPGA and DSP programmable devices to be upgraded. The infrastructure has been tested with the μ TCA-based LLRF control system. The framework uses components available in all μ TCA systems: CPU and MCH and PCIe interface available on its backplane. Currently, three AMC modules and one RTM are supported. The project takes advantage of PCIe fatpipe interface available on μ TCA backplane for transmitting firmware. Since μ TCA defines other interfaces in fatpipe region, GbE, sRIO or custom protocol can be also used for this purpose. All of the supported AMC modules use custom IPMI commands to select the active PROM revision and JTAG configuration. In the current solution the IPMI commands are not compatible between boards. Since the firmware upgrade is becoming more and more popular in xTCA systems, the IPMI commands should be standardized by the PICMG organization. The firmware upgrade solution has been successfully tested in the digital laboratory and CMTB facility at DESY in Hamburg. The PCIe hot-plug in Linux caused real problems when parameters of PCIe endpoint implemented in Xilinx FPGAs changed. The problem can be solved by standardization of used endpoints. The programming time of PlatformFlash-based devices is relatively long. It is not so critical in case of regular firmware

upgrade. However, the programming time could be crucial for 'in-system firmware upgrade' applications, e.g. software of hardware during debugging. In those cases, optimization of firmware upgrade time would be useful. The application of partial FPGA reconfiguration should decrease the time significantly. In this case, it is not necessary to reprogram the Flash memory, the FPGA can be directly updated. The proposed framework has been tested with μ TCA chassis, however it can be easily implemented in ATCA systems.

ACKNOWLEDGMENT

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

REFERENCES

- [1] PICMG, "AdvancedTCA Base specification PICMG 3.0," Dec. 2002.
- [2] —, "Micro Telecommunications Computing Architecture," July 2006.
- [3] —, "AdvancedMC Mezzanine Modul specification PICMG AMC.0," Nov. 2006.
- [4] —, "xTCA for Physics," Mar. 2009.
- [5] PICMG, "AdvancedTCA Base Specification," Jan. 2003, PICMG 3.0.
- [6] PICMG, "Micro Telecommunications Computing Architecture Base Specification MTCA.0," July 2006.
- [7] Xilinx, "Virtex-5 FPGA Configuration User Guide," August 2010.
- [8] —, "7 Series FPGAs Configuration," February 2012.
- [9] I. of Electrical and I. Electronics Engineers, "IEEE Standard for In-System Configuration of Programmable Devices," January 2003.
- [10] —, "IEEE Standard for Module Test and Maintenance Bus Protocol," 1996.
- [11] B. G. Van Treuren and A. Ley, "JTAG System Test in a MicroTCA World," in *Test Conference, 2007. ITC 2007. IEEE International*, Santa Clara, CA, Oct. 2007, pp. 1–10.
- [12] VadaTech, "uTCA Chassis with 12 AMC Slots," VadaTech Incorporated, 11540 South Eastern Avenue, Henderson, NV89052, USA, Tech. Rep., October 2008.
- [13] —, "uTCA Chassis with 12 AMC Slots," VadaTech Incorporated, 11540 South Eastern Avenue, Henderson, NV89052, USA, Tech. Rep., September 2008.
- [14] —, "uTCA JTAG Switch Module," VadaTech Incorporated, 11540 South Eastern Avenue, Henderson, NV89052, USA, Tech. Rep. ver 0.0, July 2008.
- [15] ELMA, "12 Slot Crate - MicroTCA.4 Rev. 1.0," February 2012.
- [16] Altera, "FPGA Configuration via Protocol," May 2011.
- [17] —, "Configuration via Protocol CvP Implementation in Altera FPGAs," January 2012.
- [18] W. Lie and W. Feng-yan, "Dynamic Partial Reconfiguration in FPGAs," in *Intelligent Information Technology Application, 2009. IITA 2009. Third International Symposium on*, vol. 2, nov. 2009, pp. 445–448.
- [19] W. Gao, A. Kugel, R. Manner, N. Abel, N. Meier, and U. Keschull, "Dpr in high energy physics," in *Design, Automation Test in Europe Conference Exhibition, 2009. DATE '09*, april 2009, pp. 39–44.
- [20] Xilinx, "Xilinx In-System Programming Using an Embedded Microcontroller," March 2009.
- [21] J. Hussein and R. Patel, "Low-Profile In-System Programming Using XCF32P Platform Flash PROMs," May 2008.
- [22] B. Huang, G. Naylor, G. Cunningham, O. Goudard, J. Harrison, R. Myers, R. Sharples, and R. Vann, "Embedded Linux on FPGA Instruments for Control Interface and Remote Management," in *13th International Conference on Accelerator and Large Experimental Physics Control Systems, Grenoble, France*, October 2011, pp. 1271–1273.
- [23] PICMG, "Hardware Platform Management," June 2007.
- [24] F. Ludwig, M. Hoffmann, H. Schlarb, and S. Simrock, "Phase Stability of the Next Generation RF Field Control for VUV- and X-Ray Free Electron Laser," in *Proceedings of EPAC 2006, Edinburgh, Scotland*, 2006, pp. 1453–1455.
- [25] S. Simrock, "Requirements for the ATCA based LLRF Evaluation System," in *Review of LLRF system based on ATCA standard*, 8-9 Nov 2007.
- [26] D. Makowski, W. Koprek, T. Jezynski, A. Piotrowski, G. Jablonski, W. Jalmuzna, K. Czuba, P. Predki, S. Simrock, and A. Napieralski, "Prototype real-time atca-based llrf control system," *Nuclear Science, IEEE Transactions on*, vol. 58, no. 4, pp. 1553–1561, aug. 2011.
- [27] J. Branlard, G. Ayvazyan, V. Ayvazyan, M. K. Grecki, M. Hoffmann, T. Jezynski, I. M. Kudla, T. Lamb, F. Ludwig, U. Mavric, S. Pfeiffer, H. Schlarb, C. Schmidt, H. C. Weddig, B. Yang, K. Oliwa, W. Wierba, W. Cichalewski, K. Gnizinska, W. Jalmuzna, D. R. Makowski, A. Mielczarek, A. Napieralski, P. Perek, A. Piotrowski, T. Pozniak, K. Przygoda, S. Korolczuk, J. Szewinski, P. Barmuta, S. B. Habib, L. Butkowski, K. Czuba, M. Grzegorzolka, E. Janas, J. Piekarski, I. Rutkowski, D. Sikora, L. Zembala, and M. Zukocinski, "The European XFEL LLRF System," in *International Particle Accelerator Conference, New Orleans, USA*, May 2012.
- [28] I. Rutkowski, K. Czuba, A. Mielczarek, D. Makowski, H. Schlarb, and F. Ludwig, "Vector Modulator Card for MTCA-based LLRF Control System for Linear Accelerators," in *18th IEEE-Real-Time Conference, Berkeley, CA*, june 2012, p. in print.
- [29] A. Mielczarek, D. Makowski, G. Jablonski, A. Napieralski, P. Perek, P. Predki, T. Jezynski, F. Ludwig, and H. Schlarb, "uTCA-based Controller," in *Mixed Design of Integrated Circuits and Systems (MIXDES), 2011 Proceedings of the 18th International Conference*, June 2011, pp. 165–170.
- [30] Struck, "SIS8300 uTCA for Physics Digitizer - User Manual," Struck innovative systems, Harksheider Str. 102A, Hamburg, Germany, Tech. Rep. 1.02, November 2010.
- [31] P. Vetrov, "Hardware Design Description of the Double Advanced Mezzanine Card," Deutsches Elektronen-Synchrotron DESY, Notkestrasse 85, Hamburg, Germany, Tech. Rep. ver 0.0, February 2011.
- [32] P. Gessler, M. Bock, M. Bousonville, M. Felber, M. Hoffmann, T. Jezynski, T. Lamb, F. Ludwig, G. Petrosyan, K. Rehlich, S. Schulz, P. Vetrov, M. Zimmer, C. Bohm, A. Hidvegi, D. Makowski, and K. Czuba, "Next Generation Electronics Based on uTCA for Beam-Diagnostics at FLASH and XFEL," in *10th European Workshop on Beam Diagnostics and Instrumentation for Particle Accelerators DIPAC*, Hamburg, Germany, May 2011, pp. 294–296.
- [33] PICMG, "AdvancedMC PCI Express and AS specification PICMG AMC.1," Jan. 2005.
- [34] A. Devices, "ADSP-TS201 TigerSHARC Processor Hardware Reference," December 2004.
- [35] —, "ADSP-TS20x TigerSHARC Processor Boot Loader Kernels Operation," March 2004.
- [36] N.A.T., "NAT-MCH Users Manual," March 2010.
- [37] Adlink, "AMC-1000 Mid-Size AdvancedMC Processor Module," February 2012.
- [38] W. Gao, "DPR over PCIe," May 2011.