FPGA FIRMWARE FRAMEWORK FOR MTCA.4 AMC MODULES*

Lukasz Butkowski, Tomasz Kozak, Bin Yang, DESY, Hamburg, Germany
Paweł Prędki, DMCS, Lodz University of Technology, Lodz, Poland
Radoslaw Rybaniec, ISE, Warsaw University of Technology, Warsaw, Poland

Abstract
Many of the modules in specific hardware architectures use the same or similar communication interfaces and IO connectors. MicroTCA (MTCA.4) is one example of such a case. All boards: communicate with the central processing unit (CPU) over PCI Express (PCIe), send data to each other using Multi-Gigabit Transceivers (MGT), use the same backplane resources and have the same Zone3 IO or FPGA mezzanine card (FMC) connectors. All those interfaces are connected and implemented in Field Programmable Gate Array (FPGA) chips. It makes possible to separate the interface logic from the application logic. This structure allows to reuse already done firmware for one application and to create new application on the same module. Also, already developed code can be reused in new boards as a library. Proper structure allows the code to be reused and makes it easy to create new firmware.

This paper will present structures of firmware framework and scripting ideas to speed up firmware development for MTCA.4 architecture. European XFEL control systems firmware, which uses the described framework, will be presented as example.

INTRODUCTION

The MTCA.4 standard is derived from the Advanced Telecommunication Computing Architecture. It is enhanced for Rear I/O and Precision Timing and offers a compact environment for transmission and parallel processing of large amounts of data. The mechanics and connectivity is defined by the standard PICMG MTCA.4 specification [1,2]. Main modules are called Advanced Mezzanine Carriers (AMC). Each of them can be paired with the Rear Transition Modules (RTM); front-to-rear communication and signal transfer is handled over the so-called Zone 3 region. The basic architecture follows the idea of a centralized powerful processing unit that is connected to various AMC I/O boards over several PCIe lanes, dedicated trigger lines, clock lines and platform related management lines. The AMC backplane offers also 4 ports for low-latency-links connections (eight differential pairs, full-duplex) that can reach up to 10 Gbit/s allowing the boards to communicate using serial transmission (e.g. utilizing the MGTs).

The main function of the AMC modules is to provide communication interfaces and perform digital signal processing. All I/O lines are connected to the FPGA chip on AMC board, shown in Figure 1. Depending on the application and the used AMC, different RTM, FMC or front I/O modules can be connected. Thanks to flexibility of the FPGAs, many different applications can be run on one AMC, still using the same hardware resources If the support of hardware is already provided in the code, it can be reused in many applications. The MTCA.4 firmware framework introduces an additional abstraction layer that separates the hardware dependent logic from user application logic. The framework specifies universal interfaces on this layer. This allows the same firmware and software components to be reused, irrespective of the type of the used hardware. It means that one application logic can be run on different AMC modules.

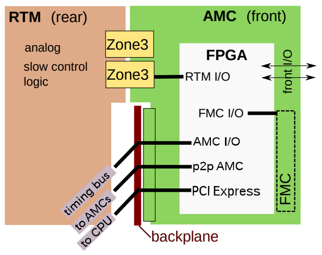

Figure 1: MTCA.4 front and rear modules

For the European XFEL all AMC boards will have one common feature: Xilinx FPGA chips, which perform, among other tasks, communication with the CPU as well as data acquisition and data processing required in control systems. All firmware algorithms are written using hardware description language (VHDL) code.

In the following sections of this paper the structure of a typical VHDL project used at European XFEL MTCA.4 [3] systems is presented.

GENERAL STRUCTURE

The code for the FPGA firmware was divided into tree main parts: board, board payload and user application.

The board section is hardware dependent and is specific for a given board. One AMC has one board support package (BSP). It has all components for configuration of peripherals on the board, such as clocks, I/O buffers, Analog to Digital Converters (ADCs), Digital to Analog Converters (DACs), etc. It is responsible for implementing communication interfaces with other boards and with the CPU. It also includes Direct Memory Access (DMA) and Data acquisition (DAQ) on DDR memory chips. The board part of the firmware provides interfaces for all resources on board.

Board payload is a middle layer section, connecting the board interfaces with the application. It contains all the

*Work supported by DESY MSK group.
#lukasz.butkowski@desy.de
ISBN 978-3-95450-148-9

possible interfaces that the selected hardware can provide, allowing the user to choose which of them will be used in the application. However, some applications may require interfaces which are not provided by a specific board. In such cases, the application algorithm needs to be adapted to the hardware or interface adapters need to be implemented.

The Application section contains user specific algorithms. This part does not interact with hardware directly and is not strongly dependent on it, although it might use interfaces which are available in hardware (board part).

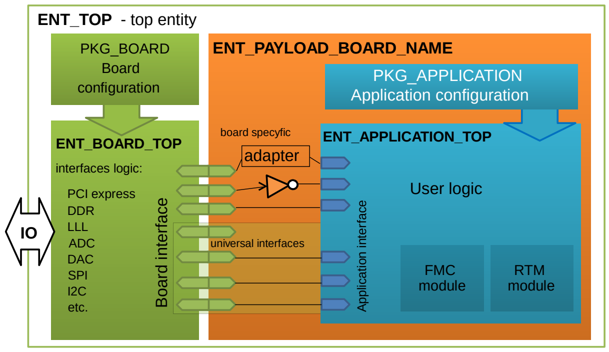

Figure 2: VHDL code block structure

Board and payload parts are connected in one top entity with all available interfaces that board can provide. One layer deeper, the application interfaces with the payload part, selecting the needed resources. Block diagram of top VHDL structure is presented in Figure 2.

INTERFACES

Between board and application we can distinguish two types of interfaces: universal and board specific. Definitions of universal interfaces are common for all boards. The include:

- **IBUS** – internal memory access bus, used to access registers and memory areas in FPGA from CPU, always connected to communication interface;
- **DAQ** – data acquisition bus, allows to write stream of data to external memory. This data is accessed using Direct Memory Access (DMA) over communication interface;
- **LLL** – low latency links bus, middle layer of point to point communication between boards using MGT;
- **FMC** – FPGA Mezzanine Card I/O;
- **RTM** – Rear Transition Modules I/O;
- **AMCIO** – backplane differential signals for clocks and triggers;
- **CLK** – clock resources.

All standard interface signals are grouped in records, which share the same name for all of the boards. This makes it easy to connect the same components in different applications. It is allowed to have more then one interface of the same type in one board. There can be a few independent DAQ channels or several point to point (LLL) connections available. In this case records are grouped in arrays.

Board specific interfaces include front I/O, ADC and DAC chips as well as any other board logic that should be controlled or accessed by application algorithms.

Communication with hardware and data acquisition is one of the most important and common task for all devices used for control systems. This is why the IBUS interface is mandatory on every application in MTCA.4 firmware framework.

Communication Interface

The data transfer between the AMCs and the CPU in an MTCA.4 crate is done using the PCI Express interface. Each board support package includes the same implementation of PCI Express transaction layer interface adjusted to the specific type of FPGA chip. It allows to access memory and memory-mapped register space on AMC board from CPU using the same abstraction layer. PCIe interconnect provides two independent IBUS interfaces. The first one connects PCIe Base Address 0 (BAR0) with board part registers address space while the second one connects PCIe BAR1 with application registers address space.

Address Space

Each VHDL module in firmware has an independent register address space definition: board logic registers, application registers and RTM, FMC, algorithms modules registers in applications. Access to those registers is done using the IBUS port on each module.

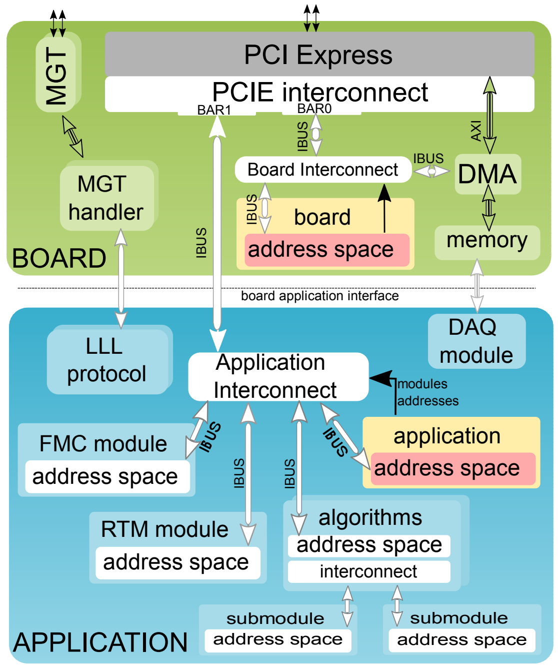

Figure 3: Block diagram of address space and interfaces connection of standard MTCA.4 firmware

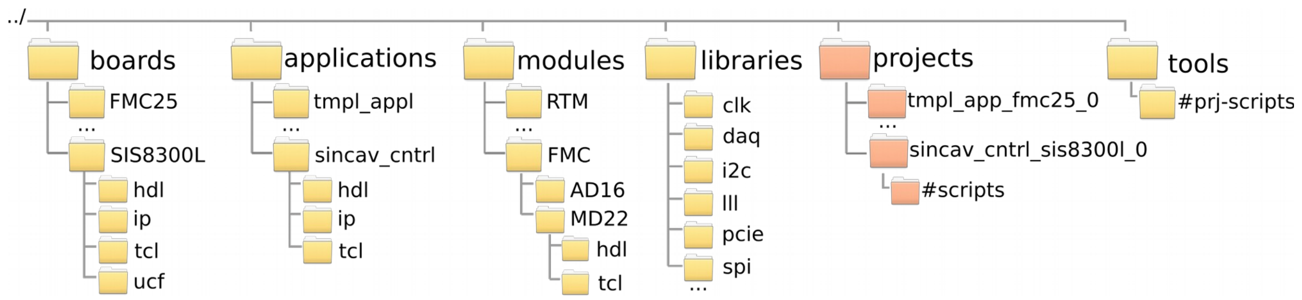

Figure 4: MTCA.4 firmware framework folder structure

This port can be directly connected to the IBUS interface from PCIe. Using interconnect many independent modules can be connected to the same bus. Information about offset of module address is defined in the board or application address space.

Main interfaces connection with modules and address space distribution is presented in Figure 3.

FOLDER STRUCTURE

Firmware files are grouped in folders based on code structure and functionality. The first level divides files into the following groups: boards, applications, modules, libraries, projects and tools. Each board support package has its own location inside the *boards* folder. The same rule applies to *application* and *modules*. Modules are additionally grouped by type of module. Structure of firmware framework is presented in Figure 4.

At the deepest level, the files are grouped by type into the following categories:

- **hdl**: all source code files written in VHDL or Verliog;
- **tcl**: scripts dedicated for source files under hdl;
- **ip**: Intellectual Property (IP) cores definition files;
- **ucf**: constraints files.

Projects folder contains the generated projects that connects all the other components together. *Libraries* as well as *modules* are shared over all projects. The tools folder contains all the general framework scripts.

AUTOMATION

The connection of board, application and modules with their configuration packages under one top entity is considered project. Project generation follows the idea that components are independent and have separate source code files list. The creation of projects is automated and done using scripts.

For firmware framework automation Tcl scripting language was chosen [4]. It is an easy, powerful and well-documented scripting language available on the majority of software platforms. It is also natively supported by Xilinx toolschain.

Tcl Scripting

Every module has its own Tcl script containing a list of source files comprising this module. This file is executed during project creation. Based on project and configuration packages variables different files can be added or additional steps can be executed, e.g. adding an automatically generated version file. Xilinx installation comes with a customized Tcl distribution called *xtclsh*. It has dedicated commands for Xilinx design flow [5].

Depending on their function, the main Tcl files included in the framework are:

- main.tcl – main Tcl script shared for all projects, main work of automation is done here, located in the tools folder;
- project.tcl – defines project constants, information about board application and its configuration packages, placed under #scripts folder of project location;
- board.tcl – defines hardware platform such as FPGA chip type and project properties, adds all board support package files to projects;
- application.tcl, module.tcl, lib.tcl – defines the user logic, modules and libraries source files to be added to the project.

Project Generation

Based on the project information and the source files lists, the project file is generated automatically. In case of the Xilinx ISE tools, a .xise file is generated. Only one command has to be executed from *#scripts* folder to perform the project generation process.

The diagram of the automatic project generation using Tcl scripts is shown in Figure 5.

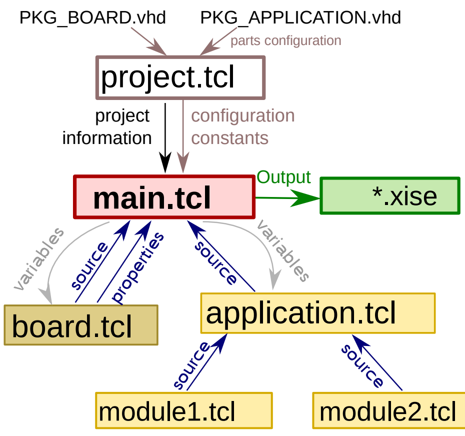

Figure 5: Project generation diagram

All the tasks are handled by the main.tcl script, which is run using Xilinx Tcl shell with the following parameters:

$xtclsh main.tcl create_project xise [ProjectName]

The script includes the projects.tcl file located in the folder given by the project name and loads the defined project variables. Those include the board and application name and their configuration packages. The packages are parsed and all the constants that are used in VHDL are accessible in the Tcl script. Base on this information, the board.tcl and application.tcl scripts are executed with access to all the variables values. Xilinx-specific Tcl commands are then used to generate the Xilinx ISE project file.

USE CASES

Presented MTCA.4 firmware framework is used for control systems at FLASH [6] and European XFEL accelerators at DESY in Hamburg. Mainly it is used at Low Level Radio Frequency (LLRF) control systems and for optical synchronization applications [7]. They are implemented using the MTCA.4 architecture. The frame-work was used throughout the entire process of development of the LLRF system. It helped and speed-up the hardware testing and implementation and evaluation of firmware.

One Board Many Applications

One of the most common cases is running a few different applications on the same board. A good example is the SIS8300L digitizer board from Struck [8]. Table 1 summarizes the main applications, in which this board is used at DESY.

The BSP code was successfully shared between the projects including different applications. Added features and bug fixed in the board part or in libraries of the VHDL code were immediately included in the all the projects, reducing the development time.

Table 1: Different Applications on the Same Board

Application	*RTM*	*AMC*
LLRF controller field detection part	DWC10	SIS8300L
Single cavity LLRF controller	DWC8VM1	
Toroid detection application	SIS8900	

One Application on Many Boards

Sometimes, an application needs to be run on different hardware than it was initially developed for. This is usually due to lack of available hardware or changes in the hardware version. This happens quite often in the hardware development process, where the prototype can differ significantly from the final version. This was the case for the main LLRF controller board. There are three versions of board with different FPGA chips and different I/O connections and all of them run the same application, as shown in table 2.

Table 2: One Application on Different Board Version

Application	*RTM*	*AMC*
Main controller application for LLRF distributed system.	DRTM-VM v1.2	uTC v.1.0
		uTC v.1.3
	DRTM-VM2	TCK7

CONCLUSION

The MTCA.4 firmware framework was developed and used at DESY. Proper structure and interfaces were defined as well as automation scripts were written. Firmware developed using this framework was successfully deployed for XFEL and FLASH accelerators. The framework by its modularity brings significant improvement in terms the development time.

ISBN 978-3-95450-148-9

REFERENCES

[1] http://www.picmg.org
[2] http://mtca.desy.de
[3] J. Branlard et al., "MTCA.4 LLRF system for the European XFEL" MIXDES, 2013 Proceedings of the 20th International Conference, pp. 109–112, June 2013.
[4] https://www.tcl.tk/ "Tcl documentation site [Online]"
[5] http://www.xilinx.com/products/design-tools/ise-design-suite.html ,"Xilinx ISE [Online]."
[6] J. Branlard et al., "Equipping FLASH with a MTCA.4-based LLRF system" SRF2013, Paris, France, pp. 1120-1122, (2013).
[7] U.Mavricˇ et al."Precision synchronization of optical lasers based on MTCA.4 electronics", IBIC2013, Oxford, UK, pp. 451-453, (2013).
[8] http://www.struck.de/ "Struck site [Online]."

ISBN 978-3-95450-148-9