COMMERCIAL FPGA BASED MULTIPURPOSE CONTROLLER: IMPLEMENTATION PERSPECTIVE*

I. Arredondo, M. del Campo, P. Echevarria, D. Belver,
L. Muguira, N. Garmendia, H. Hassanzadegan, M. Eguiraun, ESS-Bilbao, Spain
J. Jugo, V. Etxebarria, University of the Basque Country, Leioa, Spain

Abstract

This work presents a fast acquisition multipurpose controller, focussing on its EPICS integration and on its XML based configuration. This controller is based on a Lyrtech VHS-ADC board which encloses an FPGA, connected to a Host PC. This Host acts as local controller and implements an IOC integrating the device in an EPICS network. These tasks have been performed using Java as the main tool to program the PC to make the device fit the desired application. All the process includes the use of different technologies: JNA to handle C functions i.e. FPGA API, JavaIOC to integrate EPICS and XML w3c DOM classes to easily configure the particular application. In order to manage the functions, Java specific tools have been developed: Methods to manage the FPGA (read/write registers, acquire data, ...), methods to create and use the EPICS server (put, get, monitor, ...), mathematical methods to process the data (numeric format conversions, ...) and methods to create/initialize the application structure by means of an XML file (parse elements, build the DOM and the specific application structure). This XML file has some common nodes and tags for all the applications: FPGA registers specifications definition and EPICS variables. This means that the user only has to include a node for the specific application and use the mentioned tools. A main class is in charge of managing the FPGA and EPICS server according to this XML file. This multipurpose controller has been successfully used to implement a BPM and an LLRF application for the ESS-Bilbao facility.

INTRODUCTION

The utilization of the same hardware for the implementation of several applications has some advantages such as, the knowledge of how it works, its limitations, an easier maintenance and development of new tools.

Normally each piece of hardware has its own purpose but with the popularization of FPGAs the hardware has become reconfigurable. The development of an FPGA based hardware could take a long time and, therefore, the utilization of commercial hardware is beneficial. This choice contributes with a fast implementation and reliability, because commercial hardware is, usually, well tested.

In this work the VHS-ADC board of Lyrtech company [1], combined with a PC are used to design a multipurpose controller focusing their function on particle accelerators applications.

* iarredondo@essbilbao.org

In particle accelerators the most of the devices have to be connected to a central network to be controlled or monitored via the facility's standard middleware. Concretely in this paper EPICS [2] is implemented.

On the other hand, and following with the idea of the fast implementation and easy maintenance, it is important to provide some tools to make easier to the programmer the implementation of each application and to allow the user configuring the hardware without the necessity of programming. Therefore, some libraries have been designed to help the programmer and XML language has been chosen to interact with the user.

There are two main processes which are performed in the applications: Monitor and Control. Monitoring process is summarized in read data from the experiment with the hardware, access to this information and publish it on the network. On the other hand, the process of control can be described as a monitoring process with the possibility of interacting with the experiment. This is performed by changing a parameter, sending it through the network to a hardware controller and write it in the hardware.

In the following sections it will be described how to do the implementation of these two processes in a reconfigurable way with a minimal programming by the final user.

SYSTEM OVERVIEW

The main idea of the system is to use a commercial FPGA, the VHS-ADC board of Lyrtech company, to manage different applications related to control and monitor processes and data logging in large facilities. To drive the FPGA, a Host PC is utilized. This PC implements the Hardware Controller (HC) which is the link between the FPGA and the user.

The structure of the system is depicted in Figure 1. The instrument under control is connected to the Lyrtech board which is linked to the Host PC via a cPCI port. The HC reads and writes parameters from and to the FPGA's registers and publishes them over the network by means of an EPICS server. Then, EPICS clients can use the data and change the control parameters.

The structure of the conceptual programming is the following: Acquisition, fast calculations and real-time process are handled by the FPGA while slow or heavy calculations, data logging and EPICS related functions are performed in the HC.

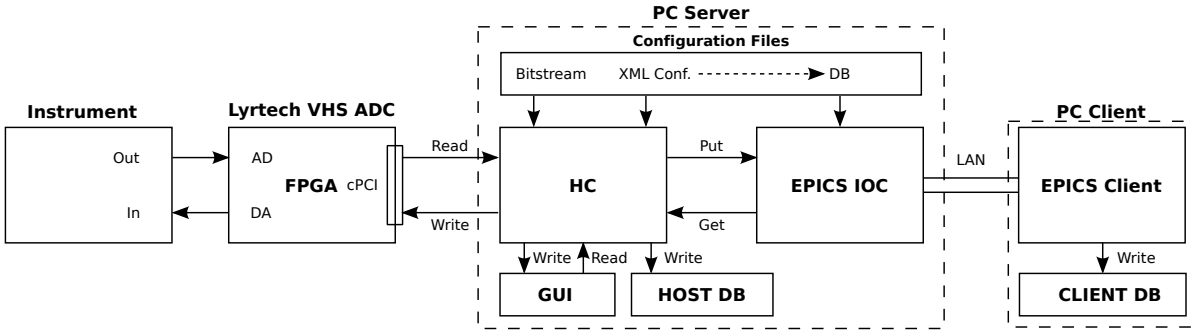

Figure 1: General diagram.

HARDWARE CONTROLLER IMPLEMENTATION

The HC is the core of the system since it is in charge of managing the FPGA, the EPICS server, the EPICS IOC, the DB, the XML configuration files, the local user GUI and of performing some calculations. The integration of all these devices has been carried out utilizing Java language because it is able to handle all of the mentioned needs. Each case is detailed as following:

FPGA: It uses a C programmed API, so JNA [3] is utilized.

EPICS Server and IOC: There is a Java based tool called JavaIOC [4].

DB: It is mainly handled by the java.sql class, because it is implemented in a MySQL database [5].

XML: The configuration file is relatively small and a fast access to the data is required. DOM technology from w3c [6] has been chosen.

GUI: SWT toolkit [7] is used.

The HC program is built with several threads in parallel, to be able to read and write from/to the FPGA, create the EPICS server, update the DB, build the GUI and parse the XML configuration file[1].

However, in order to guarantee versatility and the ease of programming different applications to the final user, some tools, xml rules and instructions have to be provided. These are treated on the following sections.

Managing Tools

In order to provide a set of tools to program the part of the HC which is related with the specific application, some Java classes which fit most common necessities have been designed. These are:

org.essb.mc.epics.db: EPICS Archiver MySQL DB implementation, managing and configuration tools.

org.essb.mc.epics.db.tables: EPICS Archiver MySQL DB formatting utils.

org.essb.mc.epics.utils: EPICS utils to manage a javaIOC: create/destroy context, connect/disconnect channels, caget synchronous/asynchronous, caput synchronous/asynchronous, create/destroy monitor and camonitor.

org.essb.mc.fpga.program: Handle the FPGA: Open/-Close board, program FPGA, program Flash, set FPGA clock, set ADCs status, read ADCs overflow and Read-/Write Registers.

org.essb.mc.fpga.maths: FPGA raw to engineering units conversion and vice versa, and standard numeric conversions.

org.essb.mc.gui.general: Utils to create the GUIs with the most common objects.

org.essb.mc.xml: Tools to acquire the configuration data from an XML document and use it in the main program.

These tools are enough to program a very wide range of applications. Nevertheless once one is set up and working, it is better to have an easy way to reconfigure some parameters depending on the specific device or situation. Here is when the XML configuration file makes the MC more versatile.

XML Based Configuration

With the XML configuration file, the goal is to reconfigure the HC to manage the desired application. Therefore, the first action taken by the HC when it is started is to ask the user for a configuration file. Then it is parsed and dumped in a well defined object in the program. All the necessary tools to perform this issue are in the org.essb.mc.xml class.

The time constraints are, normally, a common characteristic in the accelerators applications, hence, it is important to guarantee a fast access to the data of the configuration file. For this reason, it is better to have this data in the RAM and, therefore, DOM technology is very suitable.

A typical configuration file is like the following one:

```
<!DOCTYPE MCEBC [ <!-- Multipurpose
    Controller Ess Bilbo Configuration-->
  <!ELEMENT MCESSB (Name, Value, FPGA, EPICS,
    BPM)><!-- Multipurpose Controller ESS
    Bilbo-->
  <!ELEMENT Name (#PCDATA)>
```

[1] If more information about the structure of the HC is need, it could be found in [8].

```
<!ELEMENT Value    (#PCDATA)> <!--
    Initial Value -->
<!ELEMENT FPGA   (Used,Bits,Prec,Reg,
    Custom,Custom_ADDR,Signed,RW)>
  <!ELEMENT Used     (#PCDATA)>  <!-- Is
      used in FPGA ?-->
  <!ELEMENT Bits     (#PCDATA)>
  <!ELEMENT Prec   (#PCDATA)>
  <!ELEMENT Reg    (#PCDATA)>
  <!ELEMENT Custom   (#PCDATA)> <!--
      Custom => true or Lyrtech Reg =>
      false -->
  <!ELEMENT Custom_ADDR    (#PCDATA)> <!
      -- Custom register Address -->
  <!ELEMENT Signed   (#PCDATA)> <!--
      Signed => true or No signed =>
      false -->
  <!ELEMENT RW   (#PCDATA)> <!-- Read/
      Write=> 0,1 -->
<!ELEMENT EPICS (Used)>  <!-- EPICS
    config if vble is used in EPICS -->
  <!ELEMENT Used   (#PCDATA)>  <!-- Is
      used in EPICS ?-->
<!ELEMENT BPM (Used,Element,Fit)>  <!--
    BPM config -->
  <!ELEMENT Used   (#PCDATA)>  <!-- Is
      used in BPM ?-->
  <!ELEMENT Element (#PCDATA)> <!--
      Indicate which is the variable in
       BPM application. Left, Right, Up
      , Down (buttons), Amp, Phase, X
      or Y -->
  <!ELEMENT Fit (#PCDATA)> <!-- BPM
      nonlinear fitting values config
      -->
]>
```

There are five well defined fields into the XML file. The first four ones are compulsory and are repeated for all applications. These are: The name of the variable, its initial value, the FPGA's register configuration and if it is EPICS published or not.

On the other hand, there is the application field which has to be detailed by the user. In the example there is a BPM structure which has fields linking/explaining which button is measuring the variable and some values to fit the electrical delays.

The implemented data structure for the reconfigurability of the HC combines the DOM data calls, with a vectors structure which contains the index of the elements with the same characteristics. For example, in one vector it is stored the index of the elements of the DOM structure which have to be published in EPICS. The implemented structure is in Figure 2. The vectors (Value, FPGA/Read, FPGA/Write and EPICS) contain the index of the DOM node which fits with its function and, on the other hand, there are some structures which will contain the needed information associated with the vector. Conceptually, the working mode is

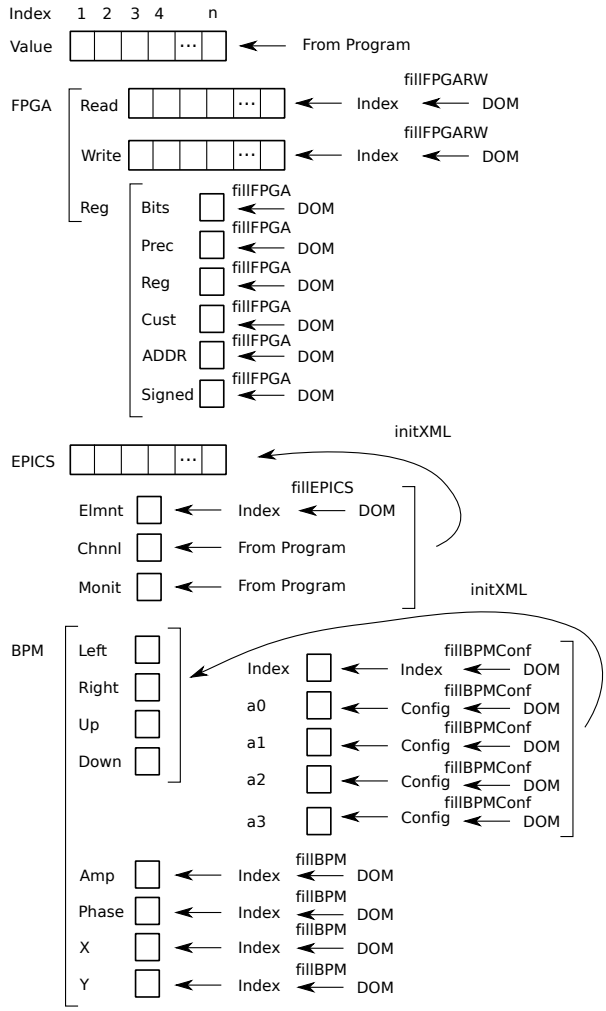

Figure 2: General diagram of communication between the control algorithm and the Host PC.

the following: Firstly the vectors are filled with the corresponding DOM node index. Then, when the program requires some data, it searches inside the vector, fills the corresponding structure with the tags of the node and uses this structure. In Figure 2 it is also detailed the method which is used to handle the data from the DOM.

Implementing a New Application

The steps to implement a new application assuming that the FPGA is suitable for the case (range of signals, ...) are the following:

1. Create the bitfile of the FPGA which fit with the application.

2. Update the configuration file:

   (a) Update the FPGA structures according to the previously designed FPGA bitfile.

   (b) Update the EPICS structures.

(c) Include a new structure for the application in the configuration file.

3. Use the org.essb.mc.xml tools to integrate the new structure into the HC.

4. Use the org.essb.mc.gui.general tools to adequate the GUI to the application. It is only needed to change the application tab, because the FPGA handling one is always the same.

CONCLUSIONS AND FUTURE WORK

In this work, a multipurpose controller based on a commercial FPGA has been presented. It has been designed to fit the large scientific facilities' requirements, using EPICS standard. Therefore, it has to be able to: Perform very fast calculations and precise acquisition, be integrated into the global control network and store the required data. The proposed solution, integrates an FPGA, an EPICS connection and a MySQL database in order to fulfill all of these specifications.

The key idea of this paper has been to explain how this controller has been made versatile, designing programming tools and using XML technology. Concretely, Java tools have been built to ease the labour of the programmer and an XML reconfiguration file has been integrated to make the final user able to configure the device.

Also it has been proposed a method to implement a new application based on the defined tools.

As future work it is foreseen the integration of EPICS database, which is an XML file, into the main XML configuration file. To perform this integration, the use of XLST is expected.

REFERENCES

[1] Lyrtech, http://www.lyrtech.com/.

[2] EPICS, http://www.aps.anl.gov/epics/.

[3] Java Native Access, https://jna.dev.java.net/.

[4] JavaIOC, http://epics-pvdata.sourceforge.net/.

[5] MySQL, http://www.mysql.com/.

[6] XML DOM, http://www.w3.org/DOM/.

[7] SWT, http://www.eclipse.org/swt/.

[8] I. Arredondo et al. "Fast acquisition multipurpose controller with epics integration and data logging", In *IPAC'11*, San Sebastian, Spain, 2011.