FAST ACQUISITION MULTIPURPOSE CONTROLLER WITH EPICS INTEGRATION AND DATA LOGGING

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

Abstract

This work introduces a fast acquisition multipurpose controller (MC), with EPICS integration and Data Logging. The application implemented in the MC system is configured by means of XML files. The main hardware is an FPGA based board, connected to a Host PC. This Host computer acts as the local controller and implements an IOC, integrating the device into an EPICS network. Java has been used as the main programming language in order to make the device fit the desired application. The whole process includes the use of different technologies: JNA to handle FPGA API, JavaIOC to integrate EPICS and XML w3c DOM classes to configure each particular application. Furthermore, a MySQL database is used for data storage, together with the deployment of an EPICS ArchiveEngine instance, offering the possibility to record data from both, the ArchiveEngine and a specifically designed Java library. The developed Java specific tools include different methods: FPGA management, creation and use of EPICS server, mathematical data processing, Archive Engine's MySQL database connection and creation/initialization of the application structure by means of an XML file. This MC has been used to implement a BPM and an LLRF application for ESS-Bilbao.

INTRODUCTION

In particle accelerators, the most of the required actions have to accomplish with remote controlling/monitoring and data logging, and some of them have other common requirements such as real-time control, fast signal acquisition, accurate synchronization or fast calculation rate. Hence, high performance and reconfigurable hardware solutions focussed on such applications are interesting.

In this paper, a Multipurpose Controller (MC), which provides a versatile tool to implement a wide variety of applications, is described. The main features of this controller, regardless the control action itself, are the EPICS [1] network integration and the data storage in two MySQL databases, one local and the other one remote. With the aim of being useful for a wide range of applications, a FPGA based hardware from Lyrtech [2], which integrates high performance data acquisition cards, has been chosen. Hence, the device is reconfigurable and can be used for fast and precise data acquisition, demanding control and data processing.

* iarredondo@essbilbao.org

In order to facilitate the use of the controller and its easy reconfiguration, a well structured main program and a set of programming tools are needed. In this case, those tasks are carried out by the utilization of Java and XML tools; concretely, several java packages have been specifically designed to implement the EPICS integration, the database handling, the XML specific tools and the access to the hardware.

The proposed solution is based on Java, because there are tools written in this programming language facilitating all project needs. Therefore, it acts as a link between all of them. Firstly, to manage the hardware API, which is programmed on C, Java Native Access (JNA) [3] is used. On the other hand, the integration of EPICS is performed with JavaIOC[4]. The local database is handled by the official JDBC (Java Database Connectivity) native Java driver for MySQL [5], while the remote one is controlled through the ArchiveEngine EPICS extension tool [6]. Finally, the Java library org.w3c is in charge of the XML document used to configure the application. Another advantage of using Java is the easy design of a user friendly GUI, since there are a lot of available tools to make it. In this case, SWT (Standard Widget Toolkit) has been used [7].

Remark that a pure EPICS design for developing the MC device is also possible. However, the integration with XML and other features would be more cumbersome in such case.

SYSTEM DESCRIPTION

There are two main devices which compose the MC. On one hand, a Lyrtech VHS-ADC board [2] and on the other hand, a standard PC.

As is schematized in Figure 1, the process to be monitored/controlled is connected to the Lyrtech VHS-ADC board, which shares the data with the PC. Hence the Lyrtech board, which is based on an FPGA, is dedicated to the acquisition and fast processing of the data and to generate control signals. The processed data is read by the PC and, if it is required, it also gives the instructions for generating the signals to be sent to the board.

However, the PC is not only responsible of the FPGA handling, but it also takes care about the EPICS server implementation and managing. Because of this, the PC application structure can be split into three main blocks, the Hardware Controller (HC) , the EPICS server and the Graphical User Interface.

In this manner, the HC drives the FPGA to read the data or to write the control signals. The calculation of the con-

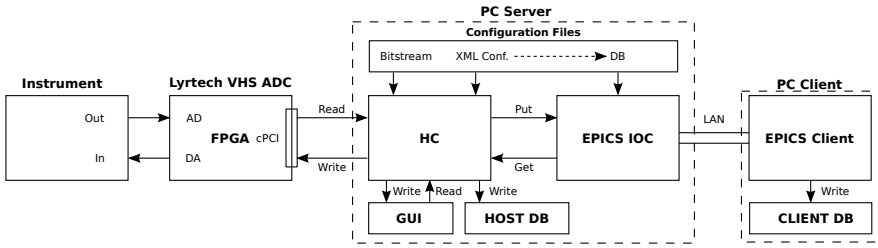

Figure 1: General diagram.

trol signals could be carried out by the FPGA, if it has to be fast, or might be performed by the HC. Even it is possible to combine both to optimize the performance.

In addition, the PC also implements an IOC server to connect the device over the network. This IOC is managed by the HC, which redirect the data to the network.

The local control and monitor of the HC is performed using an application dependent GUI. The HC writes and reads the control parameters to and from this interface.

Both the EPICS server and the HC have to be properly configured. Thus, some files, previously saved on the PC, have to be loaded.

Finally, a local EPICS client has been implemented, which is able to monitor the data and to act over some parameters to control the process. This client can be programmed in any EPICS compatible way.

MC IMPLEMENTATION

The core of the MC device is the HC block. This Java driver controls the remaining elements and the data interchange. Firstly, the main structure of the program is going to be described. Following, the Network implementation and Data Logging, which are the characteristics that become the controller multipurpose, will be treated on detail.

All the programming of the HC is performed with a set of specifically designed Java packages. These, are listed and described here:

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.

Main Structure

When the HC is launched, firstly the XML configuration file of the application is loaded (org.essb.mc.xml) and the DB configured (org.essb.mc.epics.db, org.essb.mc.epics.db.tables). Afterwards, a GUI is initialized (org.essb.mc.gui.general). This GUI has, normally, two tabs, one for the FPGA managing, which can be used by any application and another one designed for the specific application. At this moment, certain configuration data is known: the number of variables to read/write on the FPGA, the data which have to be published on the network, the signals to store and other details dependent on the usage of the MC. Also, the listeners that call the threads associated to each button/tab of the application are created.

Hence, it is possible to configure the FPGA, read and write data locally or remotely and switch on and off the DB to storage the signals.

The FPGA managing thread allows, via org.essb.mc.fpga.program package, to Open/Close the board, program FPGA and its Flash memory using the bitstream selected by the user, set FPGA clock and set ADCs status. The core of this package is the board provider's API, written in C, linked to Java through JNA.

If local control thread is activated, the registers selected in the XML configuration file are read/written (org.essb.mc.fpga.program) and presented in the GUI (org.essb.mc.gui.general) until the thread is interrupted.

On the other hand, if the remote control is selected, an EPICS server and a local EPICS client are created (org.essb.mc.epics.utils). This local client is the responsible of automatically manage the IOC to access the data (org.essb.mc.epics.utils). In order to have a local monitoring of the remote control, the values of the signals are also updated in the GUI (org.essb.mc.gui.general). Details of this process are given in subsection .

The Data Storage is started when its own thread is created. If it is required to log the data, the signals values to read or write into the registers are also saved in an EPICS Archiver format MySQL DB (org.essb.mc.epics.db, org.essb.mc.epics.db.tables). This thread can be interrupted

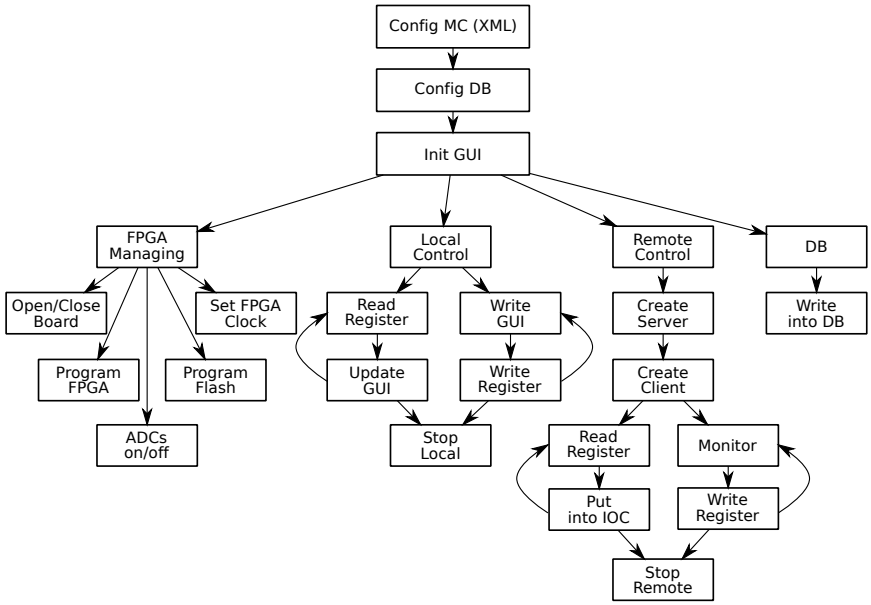

Figure 2: HC dataflow schematic

locally by the user. More information is given in section . The whole dataflow of the HC driver is in Figure 2.

Distributed Networked Control

The integration of the MC on the network is based on JavaIOC and Java Channel Access (JCA) through the org.essb.mc.epics.utils package. As has been explained in the previous subsection, firstly the EPICS server is created. This is carried out creating a JavaIOC and providing an EPICS database XML file. Once the server is ready, the HC has to interact with the IOC in automatic mode. Firstly, the context is created and, then, the channels and the monitors to access the PVs are opened. At this point, the process stands ready to read or write the PVs. This action can be performed in several ways depending on the necessities: caget, asynchronous caget, camonitor, caput or asynchronous caput. In the standard MC, the values read from the FPGA are written asynchronously to the corresponding PV and the events generated by the user are captured by a monitor and written into the FPGA. This data reading and writing is continuously repeated until the Remote Control thread is interrupted, (Figure 2).

Data Logging

Data logging is performed directly from a java thread controlled by the user, which records data as it is gathered from the FPGA, using a java precompiled SQL statement, in order to improve performance. Data is stored into a regular local MySQL database, with a particular architecture, which matches the relational database used by the EPICS tool Archive Engine. This decision was made in order to allow data storage together with EPICS data and the use of common tools for data visualization and processing. It is also worth to mention that most interactions with the database are possible via the MC, from database initialization to data visualization. Furthermore, as it was already mentioned, a second data logging tool is used, via the EPICS server which can be set up from the MC. At ESS Bilbao, the regular EPICS Achiver tool was chosen, but any EPICS client could be used.

CONCLUSIONS

In this paper, a Multipurpose Controller, which is focussed to be used on particle accelerator applications, is described. The device is prepared for be used in networked environments and for data logging, common necessities in such applications.

This MC is composed by an FPGA board with fast ADCs and a PC, which acts as Hardware Controller, EPICS server, data storage manager and user interface. Using Java and other related tools, such as JNA, JavaIOC, JCA, w3c DOM, MySQL packages and EPICS Archiver, it is possible to manage the FPGA, process the data and publish/manipulate/store it locally and over the network.

REFERENCES

[1] EPICS, “http://www.aps.anl.gov/epics/.”

[2] Lyrtech, “http://www.lyrtech.com/.”

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

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

[5] MySQL, “http://dev.mysql.com/downloads/connector/j/.”

[6] CSS, “http://sourceforge.net/apps/trac/cs-studio/wiki/RDBArchive.”

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