COMMERCIAL FPGA BASED MULTIPURPOSE CONTROLLER: IMPLEMENTATION PERSPECTIVE

Universidad del País Vasco — Euskal Herriko Unibertsitatea

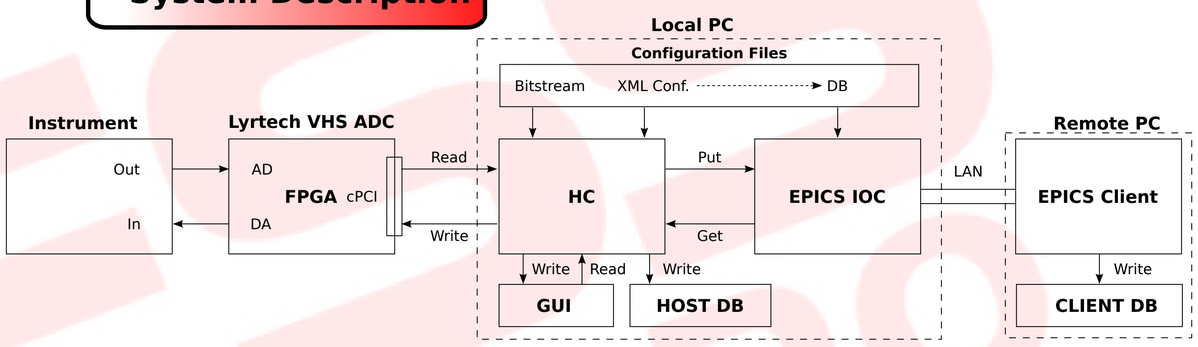
I. Arredondo*, M. del Campo, P. Echevarria, D. Belver,
L. Muguira, N. Garmendia, H. Hassanzadegan, ESS-Bilbao, Spain
J. Jugo, V. Etxebarria, University of the Basque Country, Leioa, Spain

*iarredondo@essbilbao.org

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. It is the developed main class which 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.

System Description



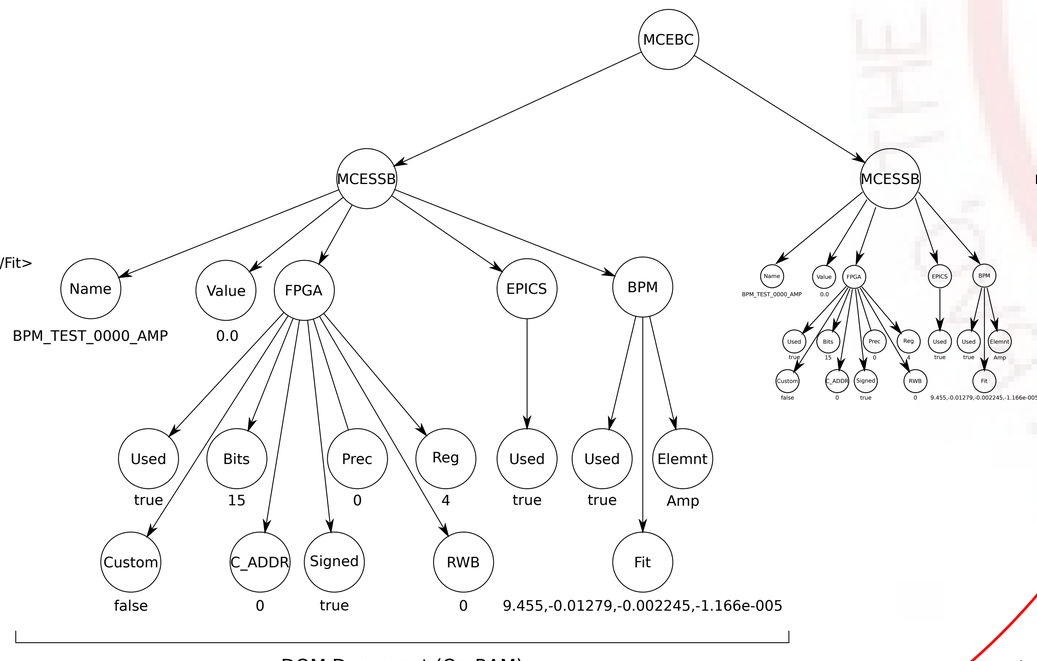
- **Instrument**: Sensor/actuator to be controlled.
- **Lyrtech VHS ADC**: FPGA based device.
  - Read data from Instrument.
  - Write data to Instrument.
  - Perform fast calculations.
- **Local PC**: Handling machine.
  - **Configuration Files**: XML and bitstream files to configure the HC, FPGA, GUI and DB.
  - **EPICS IOC**: EPICS server to communicate the data over the network. It is based on JavaIOC.
  - **Host DB**: DB to log data locally.
  - **GUI**: Graphical User Interface to allow local user control.
  - **Hardware Controller (HC)** : Java based program to handle the other devices.
    - + Configure MC from files.
    - + Read/write data from/to the Lyrtech VHS ADC.
    - + Create EPICS Server.
    - + Get/Monitor/Put from/to the EPICS Server.
    - + Read/write data from/to the GUI.
    - + Write data to the DB.
    - + Configure the FPGA.
    - + Make slow calcultaions.
- **Remote PC:** Remote PC which can handle all the functionalities remotely using EPICS connection.
  - **Client DB**: DB to remotely log the desired data.

MC Reconfiguration

Programming Tools

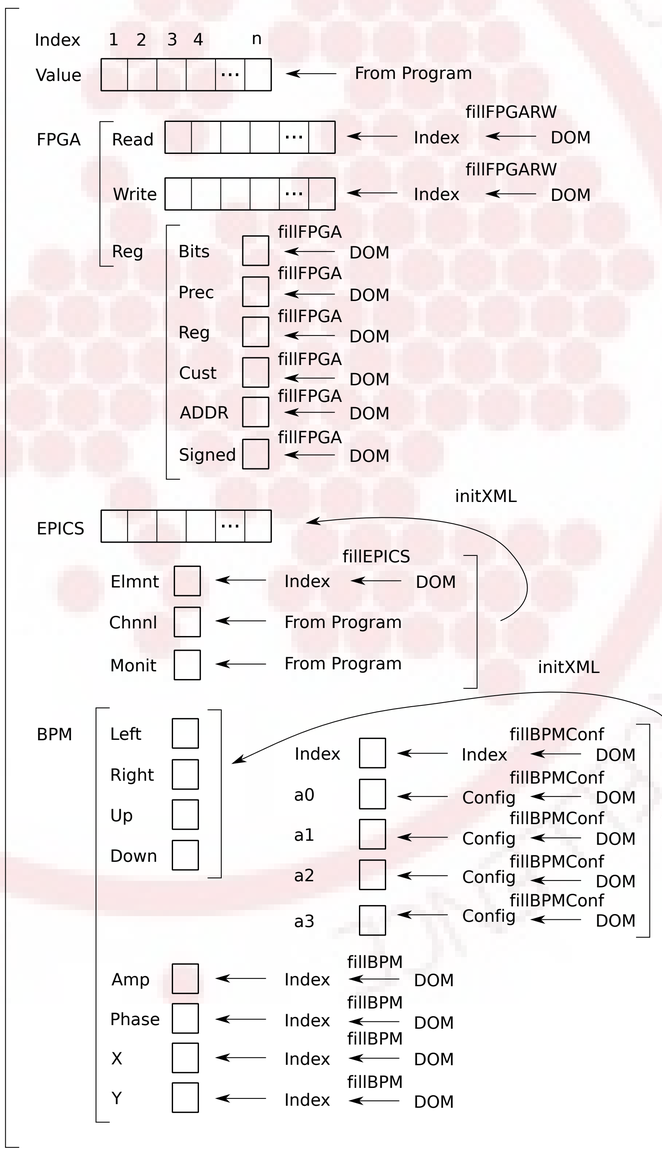
- **org.essb.mc.epics.db**: EPICS Archiver MySQL DB implementation, managing and configuration tools.
- **org.essb.mc.epics.db.tables**: EPICS Archiver MySQL DB formatting utils.
- **org.essb.mc.epics.utils**: EPICS utils to manage a javaIOC: create/destroy context, connect/disconnect channels, caget 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.

```
<MCEBC>
 <MCESSB>
  <Name>BPM_TEST_0000_AMP</Name> >
  <Value>0.0</Value>
  <FPGA>
   <Used>true</Used>
   <Bits>15</Bits>
   <Prec>0</Prec>
   <Reg>4</Reg>
   <Custom>false</Custom>
   <Custom_ADDR>0</Custom_ADDR>
   <Signed>true</Signed>
   <RW>0</RW>
  </FPGA>
  <EPICS>
   <Used>true</Used>
  </EPICS>
  <BPM>
   <Used>true</Used>
   <Element>Amp</Element>
   <Fit>9.455,-0.01279,-0.002245,-1.166e-005</Fit>
  </BPM>
 </MCESSB>
 <MCESSB>
  <Name>BPM_TEST_0000_PHASE</Name>
  <Value>0.0</Value>
  <FPGA>
   <Used>true</Used>
   <Bits>28</Bits>
   <Prec>14</Prec>
   <Reg>5</Reg>
   <Custom>false</Custom>
   <Custom_ADDR>0</Custom_ADDR>
   <Signed>true</Signed>
   <RW>0</RW>
  </FPGA>
  <EPICS>
   <Used>true</Used>
  </EPICS>
  <BPM>
   <Used>true</Used>
   <Element>Phase</Element>
   <Fit>0</Fit>
  </BPM>
</MCESSB>
```



Java Class MCEssbXML

initXML



New application implementing steps

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.