Exploiting Automatic Abstraction and the FMI Standard to build Cycle-accurate Virtual Platforms from Heterogeneous IPs

Francesco Alessandro Picchirallo - VR432225

***Abstract*—This document reports the main steps for creating Functional Mockup Unit packages, in order to be able to co-simulate a system made up of Heterogeneous IPs.**

I. Introduction

The project described consists in the realization of an FMU $Gain$ unit that has to be inserted in a complex system, respecting the FMI standard and its integration into a system made up of heterogeneous platform. The main function of the unit is to multiplies the input values by 10, the system must use data from the memory as input for the Gain component.

II. Background

The Functional Mockup Interface[1] standard defines interfaces used for the realization of a complex cyber-physical, supporting the co-simulation through the definition of an interface of the single model defined in an xml file which its functionality is expressed by a source code developed in C language. The single module implementing this protocol is called Functional Mockup Unit.

III. Applied Methodology

The project consists of two parts, since it is about of implementing the $Gain$ component and completing the $coordinator.py$ file.

A. Gain component implementation

The $Gain$ module has to multiply by 10, which is a gain constant, the input integer value. The module has the follow structure:
A $result$ variable has been added in the header file, also defined as an output port in the $modelDescription.xml$ file. If the $data\_rdy$ is true, the value has to be multiplied by the $GAIN\_VALUE$, and then saved in the $result$ variable, otherwise $result$ will be 0.

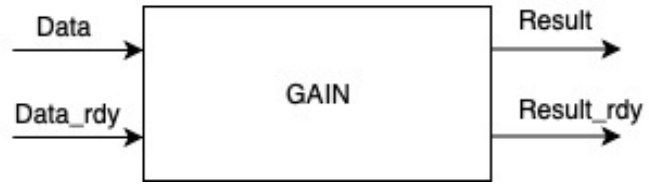

Figure 1. Gain Module

The $Gain$ module is compiled with the $cmake$ command, that generate the $gain.so$ file, used to generate the .fmu file in turn containing the XML description file and the implementation in source or binary form.

B. Coordinator

The coordinator is a the component with which it is possible to simulate the system using the FMI standard. The reference file is $coordinator.py$, which is a Python file, the framework PyFMI [2] is used to implement the master functionality in order to load the FMUs and make them interacting with each other. The functionality of the gain module has also been added to make it interact with the other modules.

C. Generation of FMUs

In this phase the various FMUs, useful for the coordinator, were generated through a script to correctly simulate the HW platform. It is sufficient to use the make command in the correct directory. The following FMU models have been compiled:

- $Accelerometer$: described in Verilog-A (analog).
- $Memory$: described in Verilog.
- $M6502\ CPU$: described in Verilog.

with the following commands:

- $verilog2hif$: used to traduce Verilog designs into the HIF format.
- $analyst$: generates discrete-event HIF models, starting from Analog designs.
- ddt: abstracts the HDL data types, replacing them with C native data-types.
- $a2tool$: generates transaction-based HIF models, starting from RTL HIF designs.
- $hif2vp$: automatically wrap an HIF model to be integrated within a Virtual Platform

Figure 2 below show the integration flow.

IV. Results

The results are visible through the output pictures, the initial value is multiplied by 10 as specified. It is possible to conclude that the functionality of the gain has been correctly implemented. Figure 3 & 4 shows input and output values.

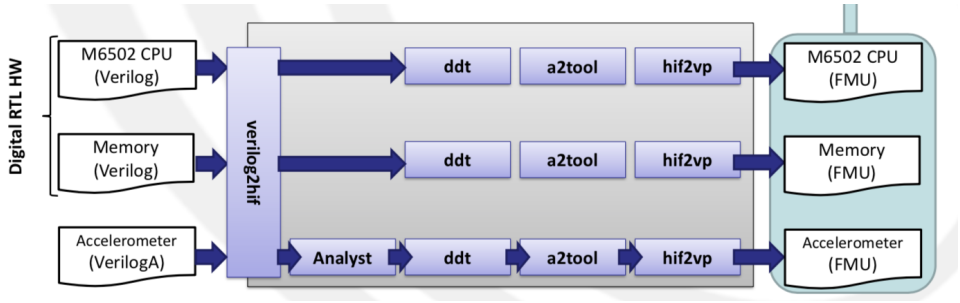

Figure 2. Integration Flow

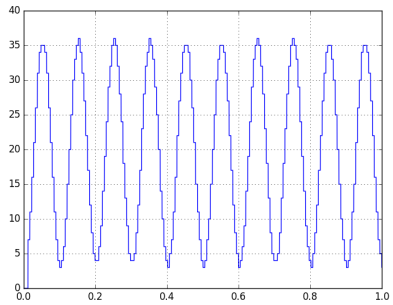

Figure 3. Input Values

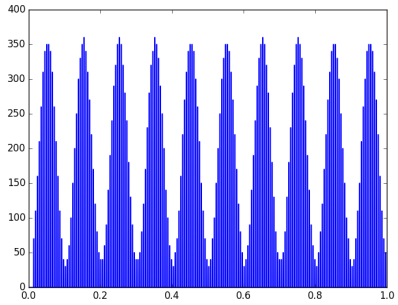

Figure 4. Output Values

V. Conclusions

The FMI standard, as seen, allows to integrate different heterogeneous components in a virtual platform to allow an easy simulation

References

[1] Modelica, "The functional mockup interface for tool independent exchange of simulation models," https://svn.modelica.org/fmi/branches/public/docs/Modelica2011/The_Functional_Mockup_Interface.pdf, 2011.

[2] M. AB, "Python pyfmi package for loading and interacting with functional mock-up units," https://pypi.org/project/PyFMI/, 2018.