FMI Add-on for NI VeriStand for HiL Simulation

Cosimo Palma Marco Romanoni
Dofware S.r.l.
10099 San Mauro Torinese (Torino, Italy)
cosimo.palma@dofware.com marco.romanoni@dofware.com

Abstract

Currently, most of the links from Modelica models to real-time hardware platforms suitable for Testing and Validation are based upon non standard model exchange format, or rely on third party preprocessor.

This paper describes the implementation of the Modelisar Functional Mock-up Interface (FMI) support in NI VeriStand, a commercial software environment suitable for real-time testing applications.

This paper presents the work conducted to implement the FMI Add-on for NI-VeriStand, which is available as a commercial product, and the process to make hardware in the loop simulation starting from a Model Based Development environment compliant with the FMI for Co-Simulation standard for model export and using it in NI VeriStand environment with National Instruments real-time hardware.

Keywords: FMI; Hardware in the Loop; NI VeriStand; Real Time Systems

1 Introduction

FMI stands for “Functional Mock-up Interface” [1] and is an open standard for model exchange specified in the ITEA2 Modelisar project [2]. The aim of this work is to enable NI VeriStand [3] to support the FMI standard for Co-Simulation. This in order to perform rapid-prototyping and hardware in the loop simulations using National Instruments hardware directly from Modelica models exported using the FMU standard. With the FMI Add-on it is possible to use FMU models in Windows and /or in National Instruments RT Targets like NI PXI [4] and NI CompactRIO [5].

In this paper, we will present:

- A description of the activity carried out for the implementation of the FMI Add-on [6] for NI-VeriStand.
- A detailed description of the steps that are to be performed in order to use FMUs in National Instruments PXI RT Targets.
- A validation test for the FMI Add-on performed with Dymola [7] and National Instruments PXI RT Target based on the detailed model of a 6 dof manipulator.

2 Scenario

Using the FMI Add-on you can make MiL/SiL/HiL with NI VeriStand framework and all Model Based Design environments compliant with the FMI for Co-Simulation standard. Figure 1 shows a typical scenario of the automotive field where some control algorithms have been designed in Simulaink and LabView, and the car model has been modeled in Dymola. With the FMI add-on and NI VeriStand it is possible to deploy the plant model along with the control algorithms in several targets and perform HiL Validation for the whole system.

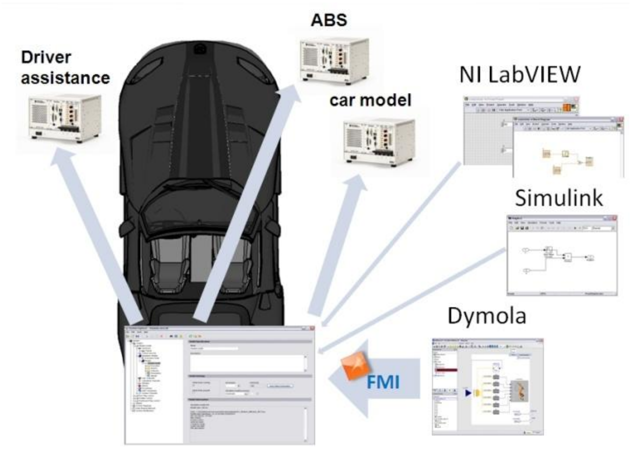


Figure 1: Example of HIL scenario in the automotive field

3 NI VeriStand

NI VeriStand is a software environment for configuring real-time testing applications. NI VeriStand helps you configure a multicore-ready real-time engine to execute tasks that include the following:

- Real-time stimulus generation
- Analog, digital, and communication bus interfaces
- Real-time stimulus generation
- Analog, digital, and communication bus interfaces
- Field-programmable gate array (FPGA)-based I/O interfaces
- Calculated channels
- Triggerable, multifile data logging
- Event alarming and alarm response routines

NI VeriStand can also import control algorithms, simulation models, and other tasks from NI LabVIEW software and third-party environments. You can monitor and interact with these tasks using a run-time editable user interface that includes many useful tools for value forcing, alarm monitoring, I/O calibration, and stimulus profile editing.

4 Co-Simulation

Co-Simulation is an approach for joint simulation of models developed with different tools where each tool treats one part of a modular coupled problem. Intermediate results are exchanged between these tools during simulation where data exchange is restricted to discrete communication points.

Between these communication points the subsystems are solved independently. Figure 2 shows an example of Co-Simulation where a complete system has been modeled using three different tools, and where each model uses is own numerical solvers and exchanges data thanks to the Co-Simulation master environment during the simulation.

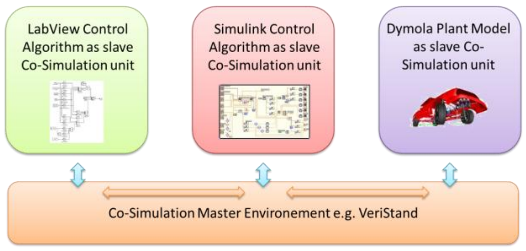

LabView Control Algorithm as slave Co-Simulation unit
Simulink Control Algorithm as slave Co-Simulation unit
Dymola Plant Model as slave Co-Simulation unit
Co-Simulation Master Environement e.g. VeriStand

Figure 2: Example of Co-Simulation scenario in the automotive field with Dymola, Simulink and LabView

5 Implementation

5.1 FMU and Real Time Target

Most Modeling and Simulation Environments compliant with the FMI 1.0 specification export an FMU file that contains an XML file and a Dynamic link library (dll) in order to maintain the intellectual properties of the model and the integrator algorithms.

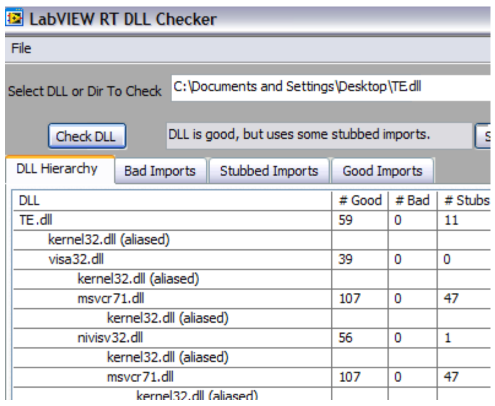
For this reason the first issue that arose in the development of the FMI add-on was how to use the model compiled as dll on RT Target running an operating system different from Windows.

Most of National Instruments Real Time Targets use Phar Lap ETS as operating system. Phar Lap ETS is a dedicated real-time operating compliant with a subset of Windows Application Programming Interface (win32 API) [8].

As consequence the dll included in the FMUs generated by the commercial modeling tools had to be checked against Phar Lap requirements.

The first tool chosen was Dymola from Dassault Systèmes, whose FMU generation routines were modified by Dassault Systèmes development team in order to solve all compatibility issues.

In order to check the compatibility of the dll included into the FMU files, LabVIEW RT DLL Checker [10] has been used. Using this tool, dll generated by any commercial or free tool, together with hand coded and compiled ones must be tested in order to check if they are compliant for Phar Lap ETS environments.


LabVIEW RT DLL Checker

File

Select DLL or Dir To Check: C:\Documents and Settings\Desktop\TE.dll

Check DLL — DLL is good, but uses some stubbed imports.

DLL	# Good	# Bad	# Stubs
TE.dll	59	0	11
kernel32.dll (aliased)			
visa32.dll	39	0	0
kernel32.dll (aliased)			
msvcr71.dll	107	0	47
kernel32.dll (aliased)			
nivisv32.dll	56	0	1
kernel32.dll (aliased)			
msvcr71.dll	107	0	47
kernel32.dll (aliased)			

Figure 3: LabVIEW RT DLL Checker window

If an FMU model passes this test, it is suitable for its use into RT Phar Lap Operative Systems such as National Instruments PXI platforms.

The implementation that is been performed to use FMUs into NI VeriStand consists in a wrapper between the FMU specifications and NI VeriStand approach to process simulation models.

We used C/C++ code for the part that has to be deployed on RT target and C# language for interact with NI VeriStand interface for acquire user specification for the model and the simulation.

5.2 Co-Simulation Master and Slave

NI VeriStand environment is a full featured Co-Simulation platform working as master for the Co-Simulation process. When several FMU's and Co-Simulation slave models exported using other model exchange formats, e.g. S-Functions, are imported into NI VeriStand, it works as master considering every imported model as slave.

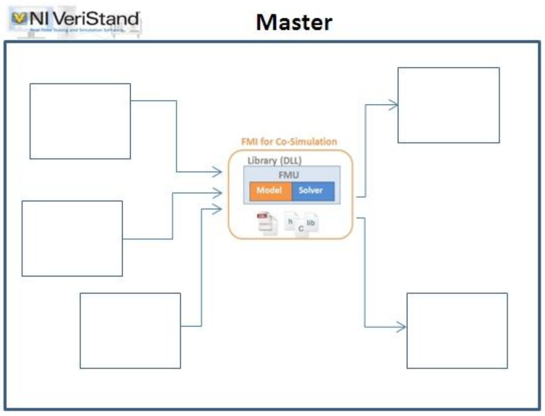


Figure 4: NI VeriStand logical schema during Co-Simulation

Data exchange between subsystems is restricted to discrete communication points. In the time between two communication points subsystems are solved independently from each other by their individual solver. NI VeriStand controls the data exchange between subsystems and the time synchronization of all slave simulation solvers.

A simulation model can be coupled if it is able to communicate data during simulation at certain time points.

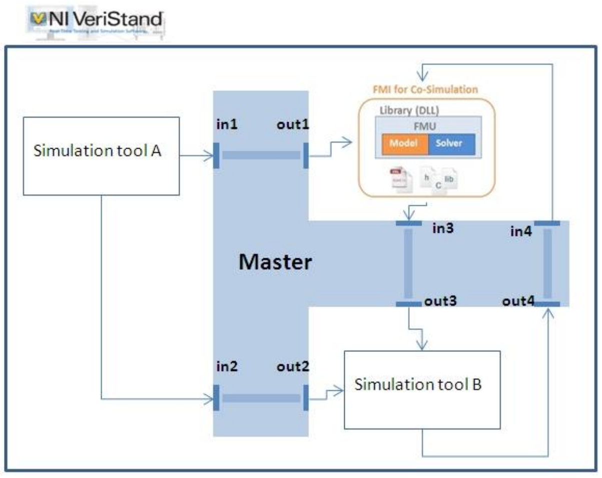


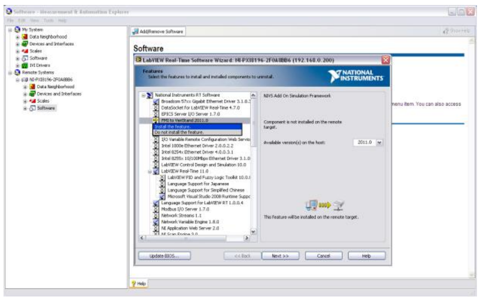
Figure 5: NI VeriStand logical schema during Co-Simulation

In NI VeriStand you can define the duration of the time step in which VeriStand will read the inputs and write the outputs. A time step is the atomic unit of time that all simulation tasks needed to be completed.

6 Using FMU models on NI Real Time

This section describes the process of validation for a model exported as FMU using an RT hardware platform.

First of all a general schema of the HIL scenario, depending on the system that needs to be validated has to be developed, see Figure 1. Once the HIL schema has been decided, the installation of the FMI Add-on must be done in each RT target that will host an FMU model, see Fig. 6.



LabVIEW Real-Time Software Wizard: NI-PXI8196-2F0A8BB6 (192.168.0.200)

Features
Select the features to install and installed components to uninstall.

National Instruments RT Software
Broadcom 57xx Gigabit Ethernet Driver 3.1.0.
DataSocket for LabVIEW Real-Time 4.7.0
EPICS Server I/O Server 1.7.0
FMI to VeriStand 2011.0
Install the feature.
Do not install the feature.
I/O Variable Remote Configuration Web Servic
Intel 1000e Ethernet Driver 2.0.0.2.2
Intel 8254x Ethernet Driver 4.0.0.3.1
Intel 8255x 10/100Mbps Ethernet Driver 3.1.0
LabVIEW Control Design and Simulation 10.0
LabVIEW Real-Time 11.0
LabVIEW PID and Fuzzy Logic Toolkit 10.0.
Language Support for Japanese
Language Support for Simplified Chinese
Microsoft Visual Studio 2008 Runtime Suppo
Language Support for LabVIEW RT 1.0.0.4
Modbus I/O Server 1.7.0
Network Streams 1.1
Network Variable Engine 1.8.0
NI Application Web Server 2.0

NIVS Add On Simulation Framework

Component is not installed on the remote target.

Available version(s) on the host: 2011.0

This feature will be installed on the remote target.

menu item. You can also access

Figure 6: Installation of the FMI add-on on NI RT targets

In order to setup the HIL system platform, each subsystem model must be exported from the model based environment in which it was developed. With the installation of the FMI add-on VeriStand will support fmu files in addition to dll, lvmodel, mdl and out files (Fig. 8) . It has to be highlighted that FMU models must be compliant with the FMI for Co-Simulation 1.0 standard. Fig. 7 shows a Dymola model exported with the FMI standard.

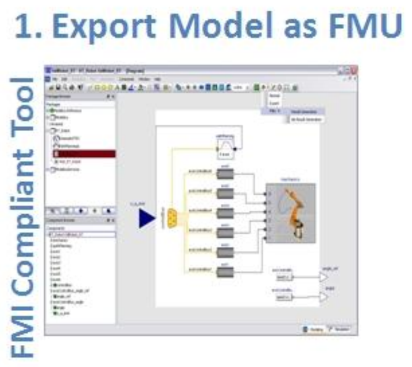
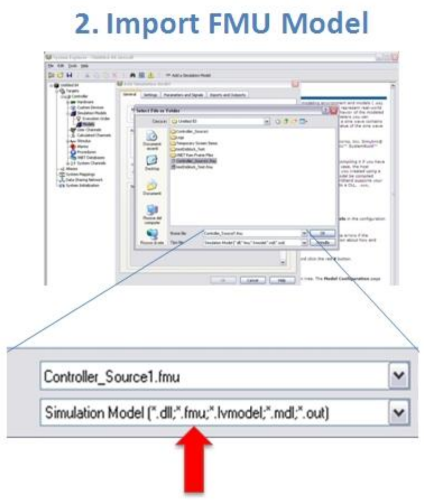


Figure 7: FMU for Co-Simulation model export from Dymola

After each sub-model has been exported, it must be imported into NI VeriStand, using the NI System Explorer as shown in Fig. 8.



Controller_Source1.fmu

Simulation Model (*.dll;*.fmu;*.lvmodel;*.mdl;*.out)

Figure 8: FMU model import using the NI System Explorer

Once the models have been imported into VeriStand, connections must be configured in order to map hardware acquisition boards I/O with model variables.

After System Explorer configuration a suitable workspace can be configured into NI VeriStand in order to control inputs and visualize the outputs in real-time, see Fig. 9. Finally the deployment on the targets will be performed automatically by VeriStand and users can use their system models in HIL.

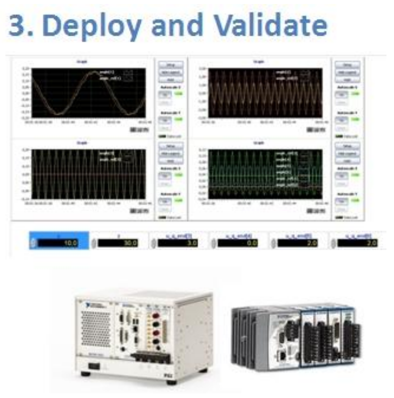


Figure 9: NI VeriStand workspace with custom scopes

7 Test and validation of the FMI add-on

In order to test the performance and validate the results of the HIL simulations performed on RT targets using FMU models, a detailed model of a six dof mechanical manipulator has been chosen as a test case. The model of the physical system is a version of the *Modelica.Mechanics.MultiBody.Examples.Systems.RobotR3.fullRobot* that can be found in the Modelica Standard Library, modified using Dymola, see Fig. 7. The original model has been modified adding one real input for each degree of freedom of the manipulator as reference angle for the motion of the electric drives. The new model can be controlled by external sources in real-time. We used also two simple control algorithms developed, one in Simulink and the other one in LabView, to control two of the inputs of the System. The purpose of this two control algorithms was to demonstrate the capabilities of this methodology and the scalability of the system architecture.

7.1 The system architecture

The architecture used to validate the system is shown in Fig. 10, and consists in three models exported in different formats from different tools that will be executed together exchanging data in real-time on a NI PXI-8196 Phar Lap RT target.

The target has been configured into VeriStand using the System Explorer, as shown in Fig. 11. Two different validation campaigns have been performed, the first using Windows and the second using Phar Lap on NI PXI-8196 as targets.

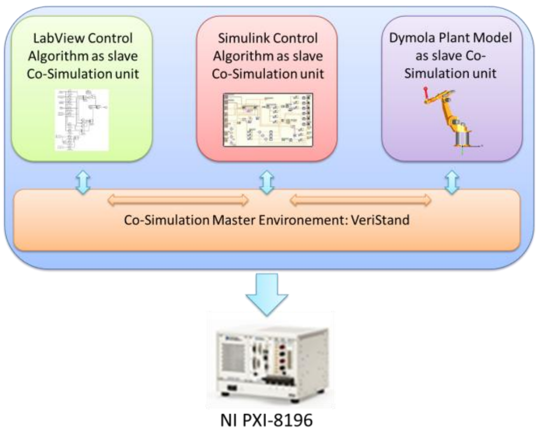

LabView Control Algorithm as slave Co-Simulation unit
Simulink Control Algorithm as slave Co-Simulation unit
Dymola Plant Model as slave Co-Simulation unit
Co-Simulation Master Environement: VeriStand
NI PXI-8196

Figure 10: System architecture for the Validation experiment

Thanks to the Dofware FMI add-on, the FMU model was loaded into VeriStand and the *depvs* file, needed for the deployment of the project into the target, automatically generated.

Thanks to the support of National Instruments development team it was possible to grant the best user experience in VeriStand to FMU importers. In fact the FMI add-on was developed in a way so that the import process in VeriStand is now the same for every model exchange format, as shown in Fig 11.

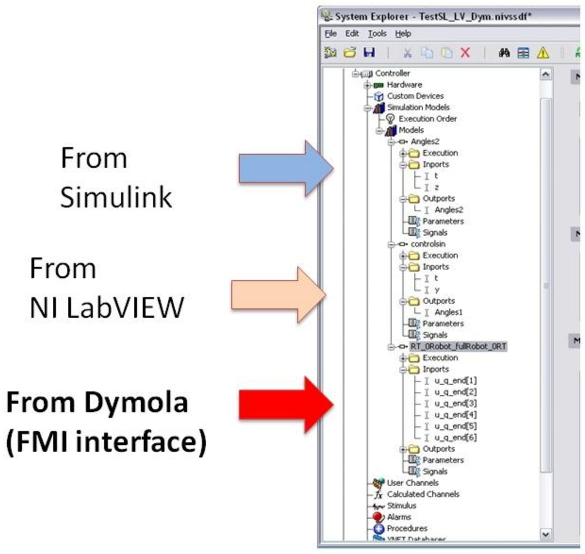


Figure 11: Import of different external models using the system explorer

It is important to note that model parameters imported using the FMU format must to be set in the modeling environment before the export phase. This is because the FMI specification does not allow parameter tuning during the simulation phase but only before the initialization phase of the model and this cannot be done into VeriStand. For this reason all the parameters of the model that need to be tunable during the HIL validation tests have to be modified and changed into inputs before the FMU generation. Still the parameters of the imported models can be seen in the system explorer as shown in Fig.12.

This issue will be more likely solved with the next release of the FMI specification that will allow parameter tuning during runtime simulations.

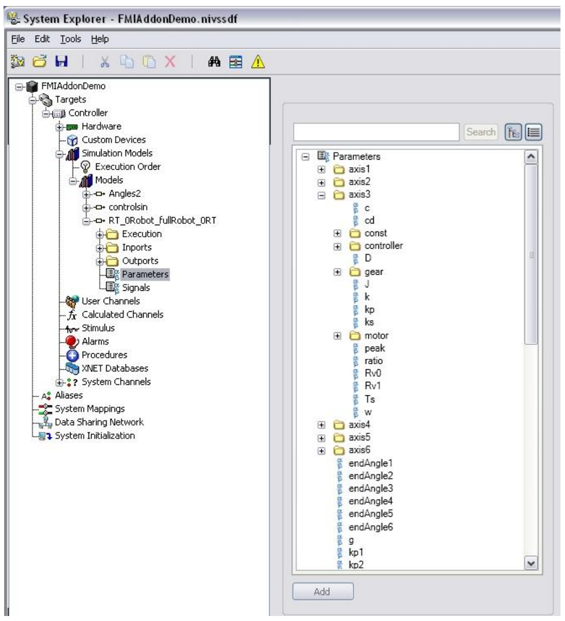
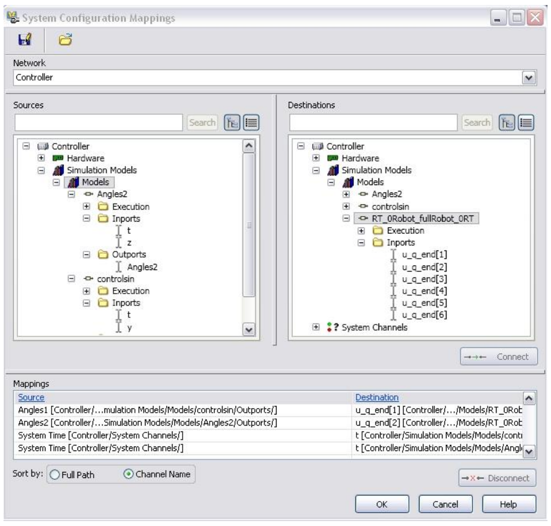


Figure 12: Model parameters explorer into VeriStand

To finish the setup phase of the system architecture, the I/O of the all models must be coupled together using the System Configuration Mappings editor in VeriStand, see Fig. 13, and with physical I/O of the target, see Fig. 14.

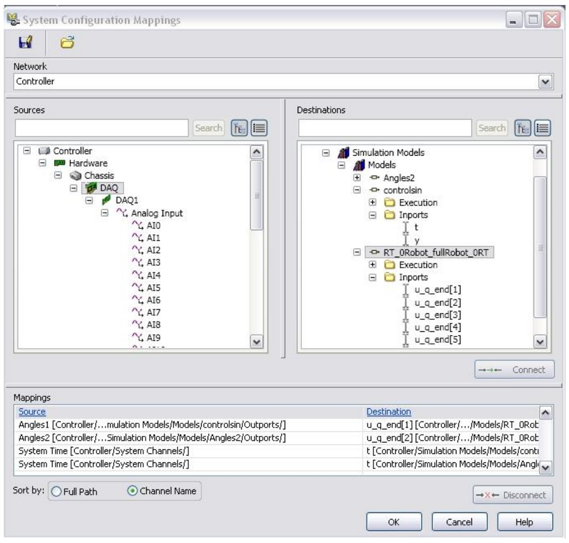


Mappings

Source	Destination
Angles1 [Controller/…mulation Models/Models/controlsin/Outports/]	u_q_end[1] [Controller/…/Models/RT_0Rob
Angles2 [Controller/…Simulation Models/Models/Angles2/Outports/]	u_q_end[2] [Controller/…/Models/RT_0Rob
System Time [Controller/System Channels/]	t [Controller/Simulation Models/Models/contr
System Time [Controller/System Channels/]	t [Controller/Simulation Models/Models/Angl

Sort by: Full Path [✓ Channel Name]

Figure 13: System Configuration Mappings editor in VeriStand, I/O of the models are coupled together


System Configuration Mappings

Network: Controller

Sources

- Controller
  - Hardware
    - Chassis
      - DAQ
        - DAQ1
          - Analog Input
            - AI0
            - AI1
            - AI2
            - AI3
            - AI4
            - AI5
            - AI6
            - AI7
            - AI8
            - AI9

Destinations

- Simulation Models
  - Models
    - Angles2
    - controlsin
      - Execution
      - Inports
        - t
        - y
    - RT_0Robot_fullRobot_0RT
      - Execution
      - Inports
        - u_q_end[1]
        - u_q_end[2]
        - u_q_end[3]
        - u_q_end[4]
        - u_q_end[5]

Mappings

Source	Destination
Angles1 [Controller/…mulation Models/Models/controlsin/Outports/]	u_q_end[1] [Controller/…/Models/RT_0Rob
Angles2 [Controller/…Simulation Models/Models/Angles2/Outports/]	u_q_end[2] [Controller/…/Models/RT_0Rob
System Time [Controller/System Channels/]	t [Controller/Simulation Models/Models/cont
System Time [Controller/System Channels/]	t [Controller/Simulation Models/Models/Angl

Sort by: ( ) Full Path [✓ Channel Name]

Figure 14: System Configuration Mappings editor in VeriStand, I/O of the models are coupled with physical I/O of the DAQ systems

VeriStand will drive the Simulation, thus the time step along with the execution order of the real time task models have to be set into the NI tool.

It is important to note that the solver's time steps of the single slave models must be smaller than the sampling time of the master device. It is also important to note that the execution time of a single time step of each single slave model must be smaller than the allocated running time in VeriStand, or the simulation will be shut down.

The allotted running time is determined by the interaction of your models and the hardware in your system. In fact VeriStand must complete all tasks in addition to the ticking of the models, such as input data processing and output data returning. The number of inputs and outputs in the system might increase the time that has to be allot for a time step. Moreover if the system includes multiple models, all of your models might need to perform a task during each given time step.

7.2 I/O Monitoring and validation results

At this point we are able to deploy the project into the target, i.e. all FMU resource files are copied to the target accordingly to the *depvs* files and the simulation can begin.

Once the simulation is running, the simplest way to monitor the state of the outputs is to develop a custom workspace into VeriStand, see Fig. 15. The workspace can include scopes for the model variables and interactive controllers for input and parameter real-time tuning.

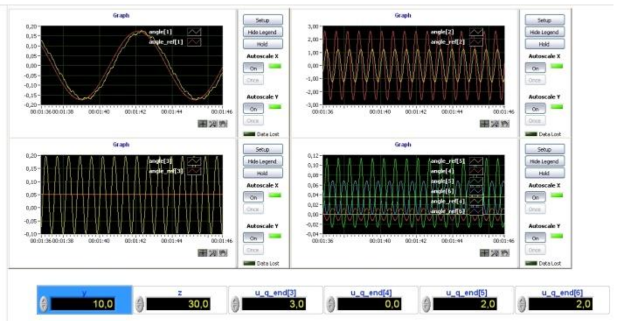


Figure 15: NI VeriStand workspace developed to control the manipulator model

Fig. 16, 17, 18 and 19 show the simulation results in terms of reference angles and simulated angles for each axis of the manipulator. The results are compared to the ones obtained simulating the same system entirely in Dymola. We can note that the simulation results in Dymola are matching with the ones obtained in VeriStand using both windows and NI PXI targets. It has also to be noted that there are some little differences from the results obtained in Dymola and the ones obtained in the PXI target generated from the "errors" introduced by different solvers and by logger used to save the output results and the limited resources of the PXI that generated data losses in the communication between the submodel units and the master sample time. The data logger was added as a custom device in VeriStand, and is capable of saving the output data on the file system using a fifo.

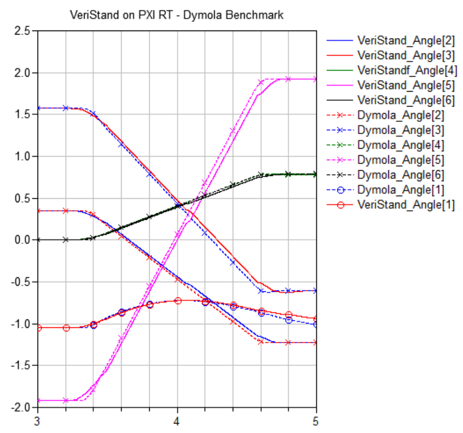


Figure 16: NI VeriStand with PXI target Vs Dymola benchmark

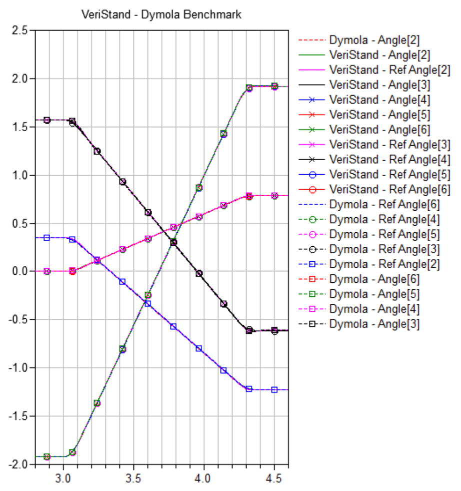


Figure 17: NI VeriStand with Windows target Vs Dymola benchmark

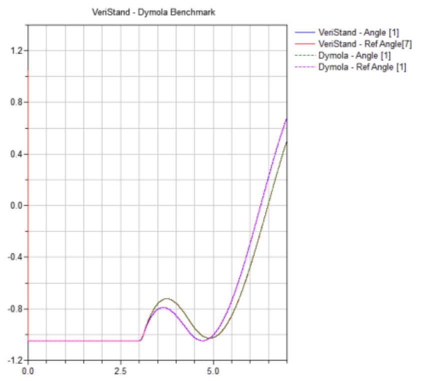


Figure 18: NI VeriStand with Windows target Vs Dymola benchmark

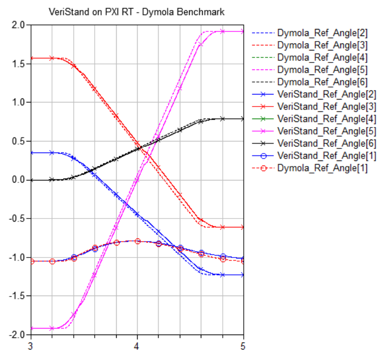


Figure 19: NIVeriStand with PXI target Vs Dymola benchmark

8 Conclusion

Thanks to the development of the FMI add-on for NI VeriStand a tool chain can now be defined to perform HIL simulation for control validation on NI hardware platforms starting from Modelica based models.

The tool chain has been validated against simulation results obtained in Dymola on a detailed system model divided into sub-systems in order to validate also the data communications between sub-models during the HIL tests. NI VeriStand present the possibility to use an external solver, so we created an addon also for FMI for model-exchange. We successfully tested it also on Phar Lap OS, but so far very few solvers have been implemented (Euler and Runge-Kutta). This solution gave us good results for simple models, but has still to be improved in order to implement solvers capable of efficiently handle complex models.

An important milestone on the roadmap of the FMI add-on will be the compatibility upgrade with respect to the FMI 2.0 specification. In this way no more modifications will be needed in order to tune the parameters of the Modelica models during HIL validations.

References

[1] Functional Mock-up Interface: http://www.functional-mockup-interface.org/index.html

[2] MODELISAR consortium. Functional Mock-up Interface for Model Co-simulation V. 1.0. http://www.modelisar.org

[3] NI VeriStand: http://www.ni.com/veristand/

[4] NI PXI: http://www.ni.com/pxi/

[5] NICompactRIO: http://www.ni.com/compactrio/

[6] FMI Add-on for NI VeriStand, http://www.dofware.com/products/fmi-add-on-for-ni-veristand/

[7] Dymola http://www.3ds.com/products/catia/portfolio/dymola

[8] FMI for Co-Simulation http://www.modelisar.com/specifications/FMI_for_CoSimulation_v1.0.pdf

[9] FMI for Model Exchange and Co-Simulation, version 2.0 http://www.modelisar.com/specifications/FM

I_for_ModelExchange_and_CoSimulation_v2.0_Beta3.pdf

[10] LabVIEW RT DLL Checker http://digital.ni.com/public.nsf/allkb/0BF52E6FAC0BF9C286256EDB00015230

[11] QTronic GmbH: FMU SDK 1.0.1 http://www.qtronic.de/en/fmusdk.html

[12] Modelica Association: Modelica – A Unified Object-Oriented Language for Physical Systems Modeling. Language Specification, Version 3.2. March 24, 2010. Download: https://www.modelica.org/documents/ModelicaSpec32.pdf

[13] Elmqvist H., Otter M., Henriksson D., Thiele B., and Mattsson S. E. Modelica for Embedded Systems. In Proceedings of the 7th Modelica Conference, Como, Italy, September, 2009.

[14] H. Hadj-Amor, C Faure, M. Ben Gaïd, N. Pernet, "Towards a Modelica Real-time co-simulation with FMI", Multiphysics Simulation - Advanced Methods for Industrial Engineering Conference, Fraunhofer, 22-23 June 2010.