Real-Time Simulator of Component Models Based on Functional Mock-up Interface 2.0

Pavel Balda
NTIS - New Technologies for the Information Society
Faculty of Applied Sciences, University of West Bohemia
Technická 8, 301 00 Plzeň, Czech Republic
Email: pbalda@kky.zcu.cz

Abstract—This paper deals with a real-time simulator of component models. The simulator imports models compatible with Functional Mock-up Interface for Model Exchange and Co-Simulation 2.0 standard (FMI 2.0) which can be generated from various versions of Modelica (e.g. OpenModelica, Dymola, SimulationX) or from several other tools. The present version of the simulator works with FMI 2.0 Co-Simulation models in which the solver is a part of the exported executable model. The simulator is integrated into RexCore – the real-time control platform. Finally, everything is compressed by the ZIP algorithm.

I. INTRODUCTION
Currently, very often appear requirements to simulate models developed in large-scale simulation systems, usually based on the ideas of Modelica, in other software environments. Examples of such environments can be various (embedded) real-time simulators, which can work in different processor platforms and other operating systems than simulation systems in which such models have been developed. So it was desirable to provide a unified interface to export models from these simulation systems in the form which is directly applicable in various target environments.

A. Functional Mock-up Interface 2.0
Probably the most famous such an interface is called Functional Mock-up Interface (FMI), whose development was coordinated by Daimler AG. The basic idea of FMI is that the dynamic system model is generated as the C language code, which can be transferred to the target simulation environment. Thus the generated model is called Functional Mock-up Unit (FMU) and is compiled using the C compiler for the target platform. Finally, everything is compressed by the ZIP algorithm into a single file with the extension .FMU.

The version 1.0 of the FMI was published in 2010 in two documents [1] and [2]. The first document describes the interface for the Model Exchange (ME), in which a model does not contain a solver. The second document describes the interface for Co-Simulation (CS) where the model of a dynamic system is generated including the solver. FMI interface began to spread rapidly, and so the improved second version [3] was released in the middle of 2014, which contain both approaches Model Exchange and Co-Simulation.

Further information on FMI 2.0 can be found in [4].

B. Current Status
Real-time simulations using Modelica models appeared even before the emergence and spreading of FMI. Such works include [5], [6], [7], [8]. Approaches to model complexity reductions are described in [9], [10]. Further increasing of the simulation performance is suggested in a more recent article [11].

Goals of introducing FMI 1.0 are described in [12]. After this event, many tools implemented FMI 1.0 import and/or export, e.g. import of FMU to OpenModelica environment [13].

At present, more than 20 simulation products or open source tools registered at FMI standard web pages implement import of FMI 2.0 compliant FMUs, the number of FMI 1.0 compliant tools is much higher. The paper [14] analyzes imperfections of FMI 1.0 and recommends quickly implement FMI 2.0. The DACCOSIM simulation system for distributed Co-Simulation based on FMI 2.0 is described in [15]. The latest paper [16] is devoted to improve FMI 2.0 CS accuracy and performance, and suggests some additions to this standard.

C. Real-Time Simulator Concept
This paper describes the development of a simulator based on import of models with FMI 2.0 interface for Co-Simulation (CS). The simulator can load a FMU generated model, write pieces of information about the model to the system log and simulate the model in real-time. The developed first version of the simulator does not care about any refinement of simulation step, this option is left on the used solver whose code is compiled and linked to the Co-Simulation model.

Three function blocks FMUINFO, FMUCS and FMUCSV described in next sections have been developed for real time simulator. All blocks utilize the freely available FMI Library [17] with the BSD license. The simulator is portable to various processor platforms that allow:

- Integrate code written in C/C++

The proper C/C++ toolchain is necessary for such platforms.

\[\text{See www.fmi-standard.org/tools for the complete list.}\]
Fig. 1. Symbols of blocks FMUIINFO, FMUCSV and FMUCS

II. FUNCTION BLOCK FMUIINFO

The function block FMUIINFO provides information on the given model in the FMU format without the necessity to analyze the internal contents of the .FMU file. The symbol of this block is depicted in Fig. 1.

The input uFMU of the block is connected to the reference of the FMU simulation model, which is the output yFMU of the simulation execution block (e.g. a block FMUCS described in next section). String outputs InNames, OutNames and ParNames contain lists of inputs, outputs and parameters of given FMU in order which is returned from functions of the used FMI Library. Individual signals are separated by the string in the parameter Separ of this block. Because typical models contain a large number of parameters (not only the model parameters in the top of the hierarchy but also parameters of all models in the lower levels of the hierarchy) the parameter SelPars allows a user to select the desired parameters that appear in the output ParNames (see table of parameters below for details).

The following subsections describe the individual inputs, outputs and parameters of the block. Names are on the left, right column, in addition to meaning of signals also indicate their types (right aligned).

Inputs

| uFMU | Reference to the instance of the FMU. |

Outputs

| iE  | Integer error code.       |
| InNames | List of input names of the FMU separated by Separ. |
| OutNames | List of output names of the FMU separated by Separ. |
| ParNames | List of parameter names of the FMU separated by Separ. |

Parameters

| SelPars | Specifies which parameters are included into the list in the ParNames output. If this string is empty (no character) the list will contain all parameters of the highest level of the model hierarchy in the FMU. Otherwise the string enumerates all parameters (of any level) that are to appear at the output ParNames, separated by , (comma). |
| Separ | String (preferably composed of a single character), which separates the names of inputs, outputs and parameters in output strings InNames, OutNames and ParNames. |

III. FUNCTION BLOCK FMUCS

The function block FMUCS allows to simulate the given FMU in Co-Simulation form (i.e. the FMU contains the model together with the solver). The symbol of this block is depicted in Fig. 1.

The parameter FMUPath stores the full path to the model file with the .FMU extension on the host (development) computer. This block supposes that the loaded model has a maximum of 16 inputs (u1..u16), 16 outputs (y1..y16) and 16 parameters (p1..p16), which are mapped to the corresponding inputs, outputs and parameters of the loaded FMU. Their exact order can be obtained using the information block FMUIINFO described in the previous section. Because the
number of the FMU parameters can be large, a user can choose only several of them by the SelPars parameter which has the same meaning as in FMUINFO.

First two inputs manage the simulation run. The Boolean input R1 serves for simulation reset to initial conditions. The rising edge (0 → 1) of this input stops the simulation, the falling edge (1 → 0) starts the simulation from initial condition in simulation time 0. The Boolean input value HLD = 1 pauses the simulation, if HLD = 0 simulation is running (including the case when the input HLD is not connected).

Nonzero value of the output iE indicates the occurrence of a simulation error. If there is an error in the simulation, the simulation is terminated until a new start of the simulation system. The Second output yFMU is a reference to the loaded instance of the FMU. This reference can be connected to the uFMU input of the FMUINFO block, which provides the information on the FMU. Disconnecting this reference from the block FMUINFO no information is collected, which may accelerate the simulation start.

The remaining parameters of the FMUCS block are used to control the simulation run. If the boolean parameter TSTOPDEF = 1, the experiment will be finished in time tstop, after which the output values do not change (as the input HLD = 1) until the next reset by R1. The parameter eps determines the required simulation accuracy.

The parameter loglevel determines the severity of messages that will be stored to the system log when running the simulation system. It is necessary to pay attention to the large number of logged messages during an ongoing simulation because it significantly slows down the simulation. It is therefore suggested to check it visually!

FMI 2.0 introduces, in addition to ordinary parameters, so called tunable parameters. Ordinary parameters can be set in the initialization phase of the model (before starting its own simulation), while tunable parameters can be changed also between simulation steps. But this requirement of FMI 2.0 specifications is not respected by some simulation systems. Therefore, the parameter TUNEALLP has been introduced.

If its value is zero, the only tunable parameters are set before each simulation step. If TUNEALLP = 1, all selected parameters (using SelPars) are considered to be tunable and the block tries to set all of them before each simulation step.

**Inputs**
- **R1**: Reset (re-initialization) of the simulation run.  
  bool
- **HLD**: Pause the simulation run (HLD=1).  
  bool
- **u1..u16**: Inputs of the block mapped to inputs of the simulated FMU.  
  double

**Outputs**
- **iE**: Integer error code.  
  long
- **yFMU**: Reference to the model instance, which is simulated.  
  reference
- **y1..y16**: Outputs of the block mapped to outputs of the simulated FMU.  
  double

**Parameters**
- **FMUPath**: Path to the FMU containing the model to be simulated. It must include the extension .FMU.  
  string
- **TSTOPDEF**: Flag indicating whether the simulation stop time tstop is defined.  
  bool
- **tstop**: Stop time of the simulation in seconds. Start time of the simulation is always 0.  
  double
- **eps**: Required accuracy of the simulation.  
  double
- **loglevel**: Severity of messages written to the system log.  
  long
- **SelPars**: Specifies which FMU parameters are mapped to the parameters p1..p16. If this string is empty (no character), all parameters of the highest level of the model hierarchy in the FMU are mapped. Otherwise the string enumerates all parameters that are mapped to the block parameters.  
  string

![Fig. 2. Model of DC Motor with Flexible Load on the Shaft](image)
The function block FMUCSV differs from the block FMUCS by using vectors for the FMU inputs, outputs and parameters which allows to use it for Co-simulation of larger models. The symbol of this block is also depicted in Fig. 1.

The majority of block signals have the same meaning as in the block FMUCS. Therefore, only different signals are explained here.

Block inputs uIn and uOut are input references to vectors containing the FMU inputs and outputs. Similarly, block outputs yIn and yOut are output references to the same vectors. The FMUCSV block does not allocate memory for these vectors, i.e. the vector referenced by uIn should contain the prepared values of the FMU inputs. Outputs of the FMU have to be processed after the execution of this block, i.e. the block(s) processing the FMU output vector should be connected to the reference yOut (more information about execution order can be also found in [18]).

The selected FMU parameters are stored in the parameters array aPar. Again, the correct order of selected parameters (inputs and outputs) can be obtained by the block FMUINFO.

The simulator configuration is depicted in Fig. 3 and Fig. 4. Parameters of the real-time executive are configured in the block EXEC (Fig. 3) which specifies task timing and priorities [19]. Period of tasks assigned to priority level 0 (connected to the output Level0) is 10 milliseconds. The block Md1Blk connected to the output Modules specifies the extension modules (.dll in Windows, .so in Linux) of the real-time executive. In this case, it is the function block library containing all FMU blocks (here FMUINFO, FMUCS and FMUCSV).

The simulation is provided in the FMUCS_task task, the configuration of which can be seen in Fig. 4. The most important block in this scheme is FMUCS_DC_motor, which loads and then simulates the model in Fig. 2. The second important block is FMUINFO, which writes the information about the loaded FMU to the system log. A part of this log is displayed in Fig. 5.

Fig. 6 shows the offline simulation of the model in OpenModelica, Fig. 7 depicts the online simulation of this model recorded in real-time by the TRND block in Fig. 4. At first sight, the excellent correspondence of both transient processes is evident. Signals differ only by rounding errors.

VI. Conclusion

The first version of a new real-time simulator based on FMI 2.0 CS has been introduced in this paper. The simulator implementation consists of three function blocks which have been integrated into real-time environment of the REX system. It is expected that the simulator will be gradually improved using the experience gained from simulation experiments from different physical domains. Especially, the approach from [16] looks promising for the future development.
ACKNOWLEDGMENT

This publication was supported by projects TA04010364 and TE02000103 of the Technology Agency of the Czech Republic and the project LO1506 of the Czech Ministry of Education, Youth and Sports under the program NPU I. The support is gratefully acknowledged.

REFERENCES


Fig. 7. Step Response in Real-Time (Online) Simulation in REX


