Synchronising a Modelica® Real-Time Simulation Model with a Highly Dynamic Engine Test-Bench System

Dietmar Winkler   Clemens Gühmann
Technische Universität Berlin
Department of Electronic Measurement and Diagnostic Technology
Sekr. EN13, Einsteinufer 17, 10587 Berlin
{Dietmar.Winkler,Clemens.Guehmann}@TU-Berlin.de

Abstract

The modeling language Modelica® is widely used by the automotive industry. In connection with Hardware-in-the-Loop (HiL) testing it can accelerate the development process enourmously. This paper presents the application of Modelica® models for a Hardware-in-the-Loop simulation using a highly dynamic engine test-bench system. Certain steps have to be taken to be finally able to connect the real-time Modelica® model to the test-bench system. One of the most important issues when connecting a simulation model to a hardware device is the synchronisation process between them. This includes the determination of interface signals, the adaption of the models according to existing interfaces, and the actually online test of the new real-time adjusted model. All these parts shall be explained in this paper.

Keywords: Hardware-in-the-Loop simulation, real-time, RT-LAB, engine test-bench system

1 Introduction

Nowadays the car manufacturers try to reduce the development times of new cars in order to cut the costs and therefore stay competitive. At the same time the manufacturer is interested in the potentials of new engine developments in terms of fuel efficiency and exhaust-gas emissions. This results in engine tests being carried out on so-called engine test-benches instead of using roller test-benches or expensive test drives. The advantage of an engine test-bench is that one does not need a prototype car into which the engine has to be mounted. A detailed model of the cars drive-train is sufficient to yield fuel saving and exhaust-gas emissions measurements from the engine test-bench. Engine calibrations can be carried out at a very early development phase using an engine test-bench for example.
Because of a cooperation of our Department of Electronic Measurement and Diagnostic Technology of the Technische Universität Berlin with the IAV GmbH Berlin, our department has access to a highly dynamic engine test-bench system. This test-bench system is used in connection with the model based calibration of electronic control units (*ECU*) of engines and transmissions [1]. As a next step object-oriented Modelica® models[1] will now be used to simulate the behaviour of all parts of the vehicle except the engine.

2 Hardware-in-the-Loop system

This section provides some more details about the applied HiL system.
The HiL system consists of a highly dynamic engine test-bench and a *HiL simulator*. The principle setup is depicted in Fig. 1.

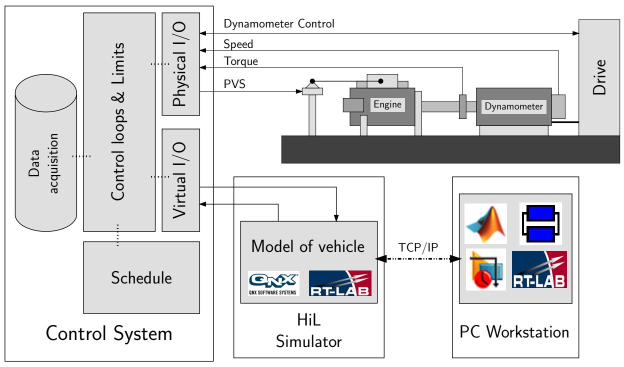

Figure 1: Principle setup of the highly dynamic engine test-bench system – “Test-Bench of the Future”

[1]Modelica® is a free modelling language developed by the Modelica Association → www.modelica.org

2.1 Test-bench system

The highly dynamic engine test-bench system consists of a combustion engine which is directly coupled with an electric *Dynamometer* which in turn is controlled by a power electronic converter (*Drive*). In addition to that a *Control System* is needed to supply the needed control signals (e.g. Pedal Value Source α for the engine, *Dynamometer Control* for the *Drive*) and to acquire the measurement data (e.g. speed and torque signals of the shaft). All this is done by the physical input/output cards (i.e. Analogue/Digital and Digital/Analogue cards).

2.2 Real-time system

The real-time system consists of a standard PC hardware running the real-time operating system *QNX*®[2] and the real-time software RT-LAB[3]. This *HiL Simulator* is connected with the *Control System* via an ether-net (UDP/IP) connection. To guarantee loss-less communication a watchdog is implemented in the simulation model.

2.3 PC Workstation

The *PC Workstation* is also a standard PC. On this PC the Modelica® simulation models are created with Dymola®[4]. These models are not suitable for real-time yet.
In order to adjust them for real-time the Dymola® model has to be included in a MATLAB®/Simulink® model[5] (this is done by using the *DymolaBlock*). The real-time software RT-LAB then automatically translates the resulting model with the *RealTimeWorkshop*®[5], transfers the C-code to the *HiL Simulator* via FTP and starts the compilation process. For another example of how to use Dymola® in connection with RT-LAB see [3]. Once that is finished the compiled model can be loaded and executed via the RT-LAB main control panel on the PC Workstation.

3 Simulation models

To demonstrate the synchronisation of a Modelica® model, we choose a standard 6-gear automatic transmission drive line model of the `Power Train Library` [4] (see Fig. 2).

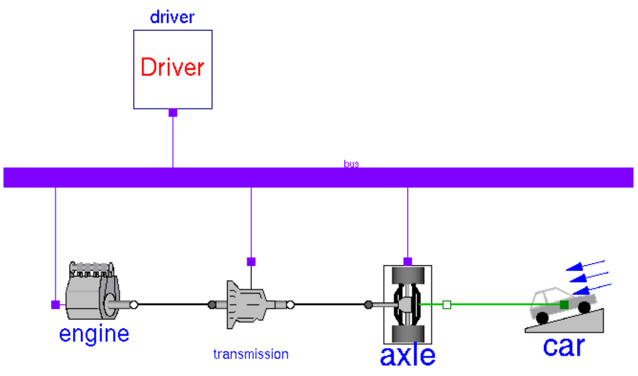

Figure 2: Graphical presentation of the drive line model

As driving cycle an excerpt of the New European Driving Cycle (NEDC) was used (The only reason for not using the whole NEDC was to speed up the simulation work, since for this work no additional information could be gained by using the complete NEDC.). Figure 3 shows the velocity in km/h over time of the NEDC excerpt when simulated offline with Dymola®. The standard Modelica® model has no input and output connectors so far. These connectors are needed when integrating the model into Simulink®. In a next step we have to define which signals the test-bench system needs and which signals the Modelica® model needs in order to function correctly as a unit. Also the exact mode of interaction of the simulation model and the test-bench system has to be defined.

3.1 Requirements for test-bench application

In our application we came across four main issues to clarify:

1. Which parts are simulated and which exist as "hardware"?
2. What control strategy is used?
3. How to synchronise the model and the test-bench?
4. What interface signals are needed depending on the control strategy?

3.1.1 Used hardware

The answer to the first question is quite simple. We want to simulate a drive line model in connection with

[2]QNX® Software Systems → www.qnx.com
[3]RT-LAB is a real-time software of Opal-RT → www.opal-rt.com
[4]Dymola® is a dynamic modelling software of Dynasim AB → www.dynasim.se
[5]MATLAB®/Simulink®/RealTimeWorkshop® is a simulation package of The Math Works, Inc. → www.mathworks.com

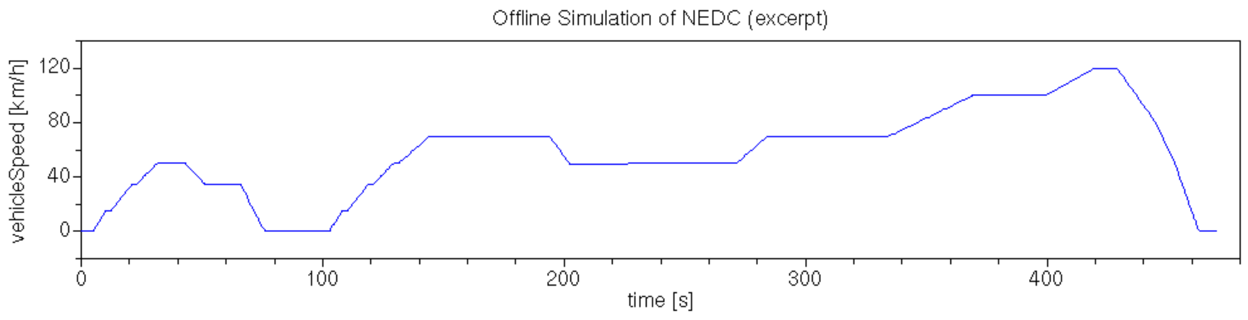

Figure 3: Excerpt of NEDC

engine test-bench system. Therefore our "hardware" device is the *engine*. So the parts which will be simulated in real-time on the *HiL simulator* will be:

- Driver (modified)
- *fake* Engine (modified)
- Transmission
- Axle (modified)
- Car

Some more details on the kind of modification applied will be given later in Section 3.2.

3.1.2 The "right" control strategy

The second question is bit harder to answer. Depending on the used test-bench system there might different control strategies available. A control strategy defines the kind of signals with which the test-bench system is controlled. In our case we have the choice of controlling the dynamometer with either a torque signal (M for moment of torque) or a speed signal (n). The engine is always controlled via the pedal value source (α). These control strategies are therefore called $M-\alpha$- or $n-\alpha$-control. Depending on the operation mode one is more preferable to the other.
In our case the drive line simulation model includes an automatic gear transmission with a torque converter. This hydraulic torque converter connects the engine outlet (pump) with the transmission inlet (turbine). A speed difference of the pump and the turbine of the converter yields a transfer torque and vice-versa. So whenever there is a torque on the load side (transmission) the inlet speed of the torque converter can be computed. So we can use the speed signal to control the test-bench system (the dynamometer to be more precise) at all times of simulation.
If we would use a manual shift gearbox a speed signal for controlling the dynamometer of the test-bench can not be obtained so easily for certain modes of operation. One of these modes being the shifting of the gear. When a gear is shifted the clutch (which is located between the engine and the transmission) is opened. When the clutch is opened it is difficult to determine the speed of the "free spinning" engine side of the clutch. To do this correctly some parameters such as the internal friction and the inertia of the engine have to be known in detail. Obtaining these parameters is not very easy (if not sometimes impossible) and comes with a lot of measureing effort. On the other hand the applied load torque of the engine side of the clutch during gear shift is known to be zero (this is true when neglecting the inertia of the clutch for now). So by using the $M-\alpha$-control strategy whilst shifting the gear we can overcome the uncertainties of not knowing the correct control speed (as needed by $n-\alpha$-control). Now the reader may ask why not using $M-\alpha$-control during all states of operation?
One reason for this is the safety issue that it is always good to know and to check the speed set-point rather than handing over a torque set-point from which a acceleration or deceleration will result. A too big torque request and hence a too big acceleration might lead to speed too high for the test-bench system even before the system itself can react. Obviously things like this can be avoided by an appropriate design of the test-bench system. But in most cases this leads to a very complicated control structure. So in practice one uses the $M-\alpha$ control strategy for modes where $n-\alpha$-control is not suitable. See [1] for more on this topic.

3.1.3 How to synchronise

Section 2.2 described in short the coupling between the test-bench system and the real-time computer on which the drive line model is running. We will give a

short example to get a better understanding of how the whole synchronisation process between the test-bench system and the drive line model works.
At first the the test-bench engine is started and put into idle mode (this can differ if someone is interested in measurement results of a “cold-start and run” cycle). In our example the engine is now running in idle speed and controlled by the control software of the test-bench system. In the meantime the drive line simulation model was loaded into the HiL-simulator (see again Fig. 2). For now both the test-bench system and the HiL-simulator are already communicating with each other via UDP. But until now they are just “listening” to each other. The next step would be to connect the simulation model to the test-bench system so that the drive line model controls the pedal value source α of the real engine and the set-point of the torque or the speed (depending on the control strategy) of the dynamometer. But before this connection can be made the difference of the control signals (i.e. α, n, M) between the measured values of the test-bench system and the calculated values of the simulation model may not exceed a certain boundary. To ensure this we use the Boolean signal `syncTBS`[6] which when activated forces the drive line model to accept the engine torque and engine speed as input set-point for the torque converter (or clutch in case of a manual shift transmission). This way the drive line model is running at the same speed like the engine regardless of the measured input torque from the engine.
Once the engine and the drive line model are running at the same speed the drive line model can take over the control of the engine. This is done by a software switch of the test-bench system. After setting the `syncTBS` signal to “false” the *HiL simulator* controls now the engine via the pedal value source α and the speed signal n or the torque signal M, depending on the control strategy.

3.1.4 Interface signals

Up to now we should have identified all necessary interface signals. These depend also on the used control strategy. To keep a certain degree of freedom in terms of which control strategy is used, we choose to supply interface signals suitable for both, $M-\alpha$- and $n-\alpha$-control. We also need some extra signals to control the Dymola® model embedded in the real-time environment. These extra signals are the signal to start the driving cycle, the signal to switch between the two control strategies and the signal to switch the drive line model into synchronisation mode.
It follows a list of the the inputs used (extra signals are marked with ‘*’).

Inputs to the simulation model:

- speed of the test-bench engine $[rpm]$
- torque of the test-bench engine $[Nm]$
- start of driving cycle $[boolean]$*
- $n-\alpha$-control active $[boolean]$*
- synchronize model to test-bench $[boolean]$*

As outputs again we needed some additional signals to display the current state of the simulation model (again marked with ‘*’). These are optional and not required by the test-bench system.

Outputs to the test-bench system:

- drive line speed $[rpm]$
- drive line torque $[Nm]$
- pedal value source α [p.u.]
- vehicle speed $[km/h]$*
- cycle speed set-point $[km/h]$*
- selected gear*

3.2 Modification of simulation models for HiL

The previous section showed the interface signals necessary to couple the *HiL simulator*(drive line model) to the test-bench system. Now this section will give some more more details in which way the drive line model had to be adapted in order to be able to provide the interface signals or to react accordingly to them.
The `Driver`, the `Engine` and the `Axle` models had to be modified:

3.2.1 Modifications to the driver model

The Boolean signal `startCycle` had to be added in order to start the driving cycle at a given time. This means that the original `CombiTimeTable` block from the Modelica Standard Library (MSL 2.2.1) had to be replaced by the two blocks `Timer` and `CombiTable1Ds` (see Fig. 4).

[6]= **synch**ronise**T**est**B**ench**S**ystem

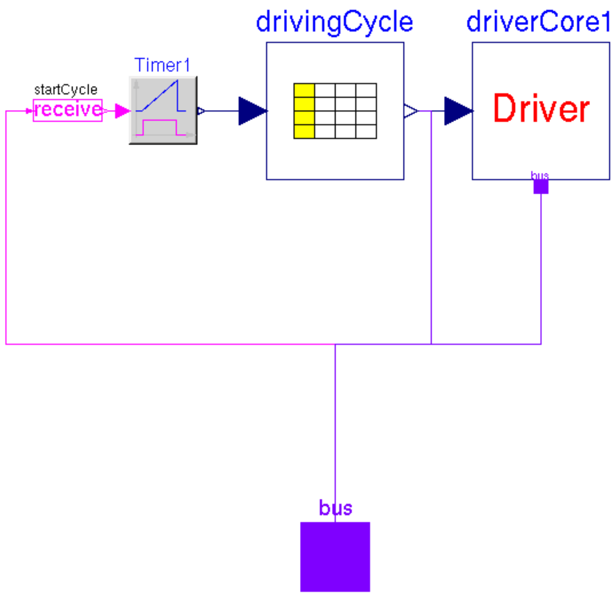

Figure 4: Picture of TriggerdDriver model

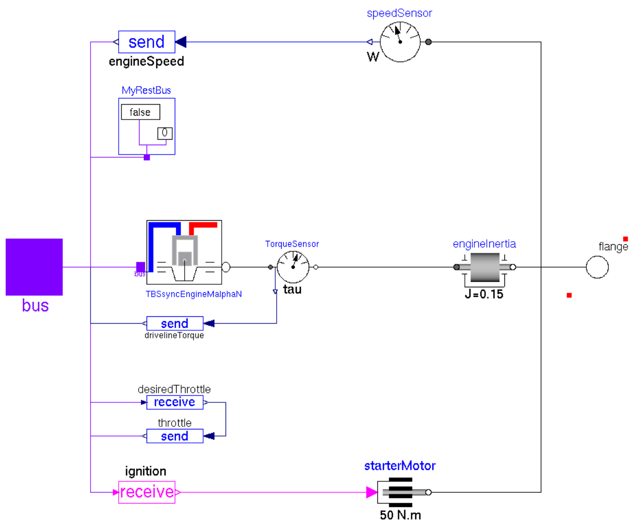

Figure 5: Picture of the FakeEngine model

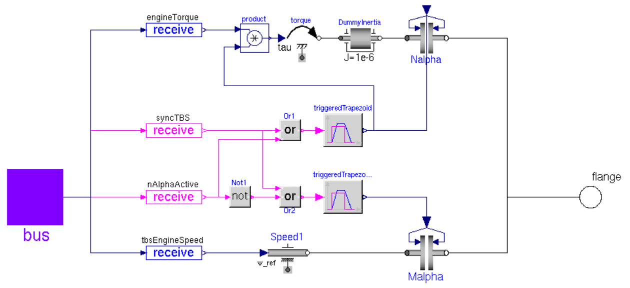

Figure 6: Picture of the TBSsyncEngineMalphaN model

Finally the driving cycle table had to be adjusted as well because CombiTable1Ds only accepts monotonically increasing data vectors. The Timer block on the other hand accepts two different function values at one time instant.

3.2.2 Modifications to the engine model

Since the engine exists as hardware part within our HiL simulation we only need some kind of “fake” engine. This “fake” engine will be used as interface to the torque or speed signal coming from the test-bench system. We did not want to break with the drive line architecture of the Power Train Library. Therefore a FakeEngine model was created based on the PowerTrain.Engine frame (see Fig. 5).
The governor had to be removed since the test-bench system includes an idle-speed control of its own. Also the torque measurement of the drive line torque is done in this model.
The FakeEngine model includes the subcomponent TBSsyncEngineMalphaN instead of the original BaseEngine subcomponent (see Fig. 6).
In the model TBSsyncEngineMalphaN all the synchronisation and control strategy functions are implemented.
When the signal syncTBS is active both coupling clutches Malpha and Nalpha are closed making sure that the drive line synchronises to the test-bench system. When the synchronisation process is complete the syncTBS signal can be deactivated. Depending on the state of the signal nAlphaActive either the Malpha or the Nalpha clutch stays closed after deactivating syncTBS. To avoid sudden changes (which in turn can cause instabilities) the closing and opening process of the clutches is done via triggered ramp signals.
The engineTorque signal is multiplied with zero as long as the $M-\alpha$ control strategy is activated. This is to avoid a constant speed-up of the left hand side of the then open $n-\alpha$ coupling clutch.

3.2.3 Modifications to the axle model

When simulating the drive line model with the CarResistance2 component of the Power Train Library included we noticed a phenomenon. At the beginning of the simulation the CarResistance2 model calculates a small erroneous torque which is applied at the axle flange. This in turn causes a small deceleration of the vehicle model (i.e.the car rolls backwards). This backward rolling behaviour causes the real-time HiL model to become unstable. A quick solution to avoid this was to activate the brakes as long as the driver is not starting to drive (see Fig. 7).

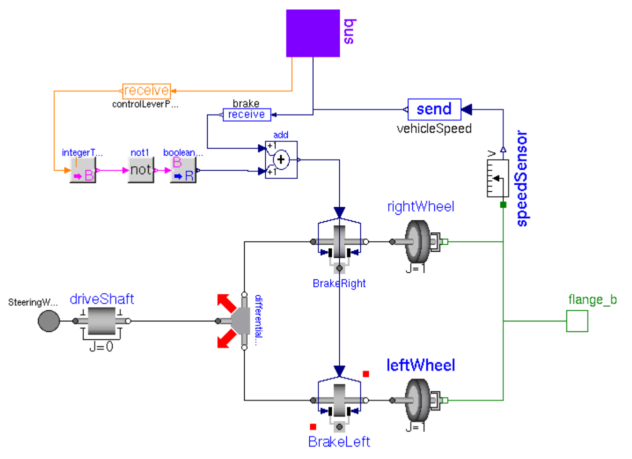

Figure 7: Picture of the `ModifiedAxle` model

4 HiL simulation results

Finally after all necessary interface signals have been defined and the modification to the drive line model has been done it is time to test the model with the test-bench system. In order to generate real-time capable code the Dymola® model is included in a Simulink® model as a wrapper using the *DymolaBlock*. The resulting Simulink® model is then arranged to fit the real-time structure of RT-LAB®. More on one how to do a real-time simulation using Dymola® in connection with RT-LAB® can be found in [5].
Figure 8 shows the velocity in km/h and the engine speed in rpm over time of the NEDC excerpt when simulated online. By online we mean the Dymola® model runs in real-time on the *HiL simulator*and communicates with the test-bench system via the UDP/IP interface. Since we are using a automatic gear transmission the $n - \alpha$ control strategy was used during the simulation (i.e. `nAlphaActive=true`).

4.1 Synchronisation

To demonstrate the synchronisation process a plot of roughly the first 30 seconds is displayed in Fig. 9.
Prior coupling the *HiL simulator* and the test-bench system together we need the drive line speed of the simulation model and the engine speed of the test-bench to be the same. We can divide process in Fig. 9 into three phases.

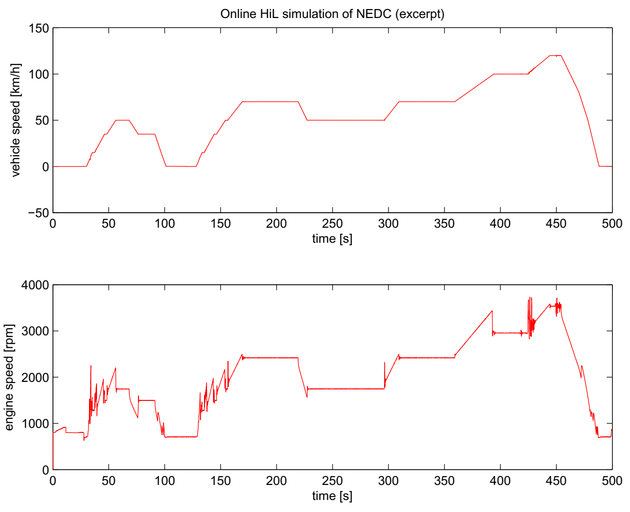

Figure 8: Online simulation of NEDC cycle (excerpt)

Phase 1: At the start the `syncTBS` signal is false. This means that the drive line model receives the torque signal of the test-bench engine. As mentioned before the test-bench engine is idling at the beginning. The torque is something near zero *Nm*. This small torque causes now the free-spinning torque converter (control lever position is still “Neutral”) to accelerate slightly (see *driveLineSpeed* in Fig. 9).

Phase 2: After about 10 seconds the `syncTBS` signal is activated. Now the drive line model is forced to the speed of the test-bench engine. Within this phase the test-bench engineer also activates a switch on the test-bench system to hand over the control to the *HiL simulator* (i.e. the coupling process is completed and n and α are now provided by the drive line model).

Phase 3: We can now deactivate the `syncTBS` signal once the two systems are coupled. Everything is now ready to start the driving cycle (i.e. `startCycle=true`). The synchronisation process is therefore complete.

5 Conclusions

In this paper we have demonstrated the process of synchronising a standard Modelica® model to a highly dynamic engine test-bench system. Different interface signals had to be added to the original simulation model so it can communicate with the test-bench system. This led to an adaptation of the simulation models

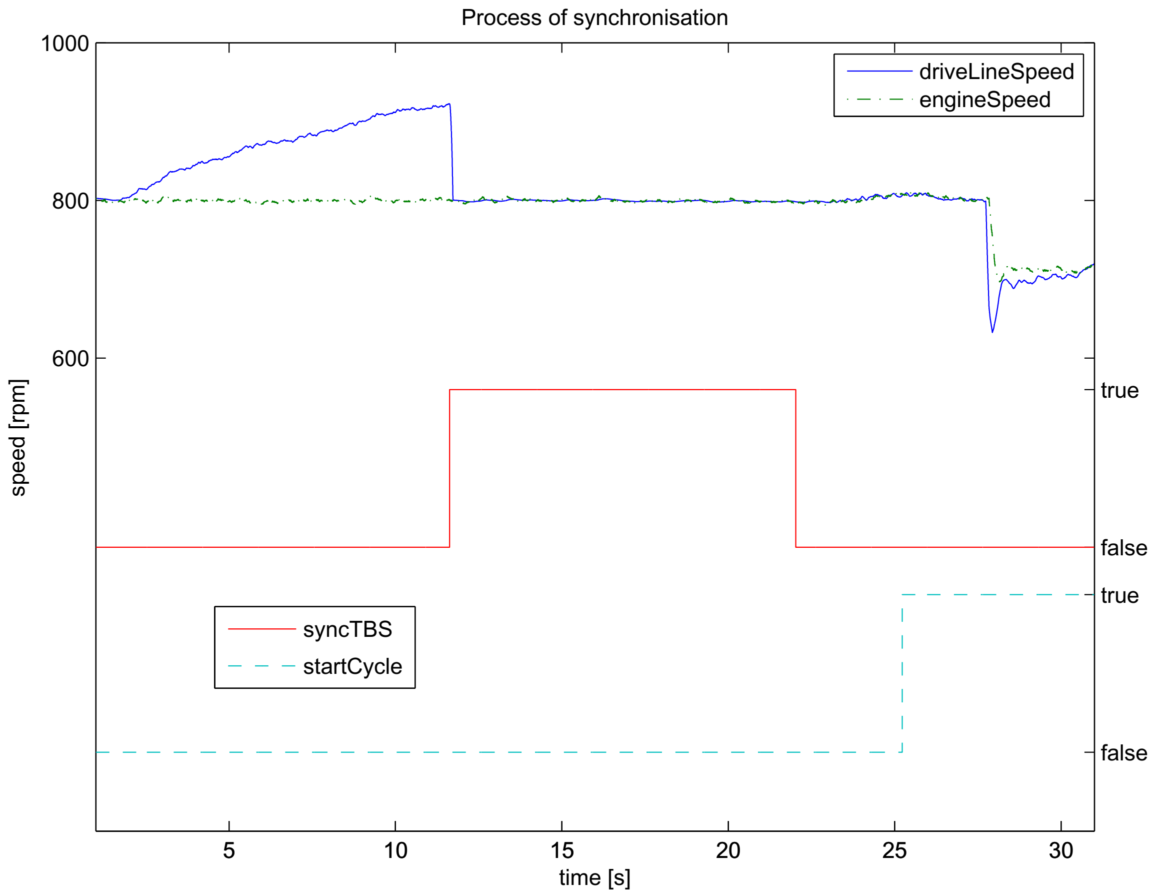

Figure 9: Synchronisation process in detail

which was presented in detail. In the end the simulation results of a working synchronisation process were shown.

The test-bench system can be used to simulate a variety of drive line models. Starting from conventional drive lines up to drive lines with a double-clutch transmission and even a hybrid electric vehicle (HEV) drive line. At all times the switching between the two different control modes (i.e. $n-\alpha$ and $M-\alpha$) is a very delicate issue especially when done “online”.

Our Department will continue to investigate the challenges of HiL simulation in connection with the engine-test bench. The focus lies hereby on the simulation of hybrid electric vehicles power trains. The test-bench system gives here the opportunity to gain measurement data of fuel consumption and exhaust-gas emissions for different HEV applications.

References

[1] D. Winkler, C. Gühmann, B. Barzantny, and M. Lindemann, “Model Based Calibration of ECUs Using a Highly Dynamic HiL Test Bench System,” in *Design of Experiments (DoE) in Engine Development II* (K. Röpke, ed.), vol. 49 of *Haus der Technik Fachbuch*, (Berlin), pp. 268–277, Haus der Technik Essen, expert verlag, June 2005.

[2] Modelica Association, “Modelica is a free modelling language.”

[3] H. Elmqvist, S. Mattsson, H. Olsson, J. Andreasson, M. Otter, C. Schweiger, and D. Brück, “Real-time Simulation of Detailed Automotive Models,” in *Proceedings of the 3rd International Modelica Conference*, pp. 29 – 38, 2003.

[4] M. Otter, C. Schweiger, and M. Dempsey, *PowerTrain Library 1.0*. German Aerospace Center (DLR), Oberpfaffenhofen, 1.0 ed., 2002.

[5] Dynasim AB, *Dymola 5 - User Manual*. Lund, 2004.