HETEROGENEOUS MODEL INTEGRATION AND VIRTUAL EXPERIMENTATION USING XMOD: APPLICATION TO HYBRID POWERTRAIN DESIGN AND VALIDATION

Mongi Ben Gaid, Gilles Corde, Alexandre Chasse, Bruno Léty, Rodolphe De La Rubia, Mohamed Ould Abdellahi

Technology, Computer Science and Applied Mathematics Division, IFP, FRANCE

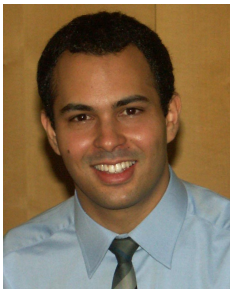
mongi.ben-gaid@ifp.fr(Mongi Ben Gaid)

Abstract

The design, development and validation of vehicle powertrains is performed in concurrent cycles, involving various teams, working on a wide range of fields, and relying on modeling and simulation. However, there are still limitations that reduce models exchange and exploitation. After reviewing the main limitations of current simulation tools and methodologies, this paper introduces a new software platform, code named xMOD, and ambitioning to improve models exchange and exploitation. The main idea of xMOD is to combine in the same platform, a heterogeneous model integration environment, as well as a virtual instrumentation and experimentation laboratory. xMOD allows model developers, on one hand, to make their models easily understandable and exploitable, thanks to its virtual instrumentation functionalities, and to protect any confidential know-how. On the other hand, xMOD makes it possible for engineers and scientists, regardless of their experience and proficiency in a given modeling language or environment, to rapidly and easily integrate and exploit heterogeneous models. xMOD enables an increase of simulation speed due to its ability to exploit multi-core processors. Finally, the application of xMOD to the design and validation of a hybrid vehicle is presented, illustrating its functionalities and demonstrating its effectiveness.

Keywords: Model integration, virtual prototyping, virtual experimentation, co-simulation

Presenting Author's Biography



Mongi Ben Gaid received the Dipl-Ing. degree in electrical engineering from the National School of Engineers of Tunis, Tunisia, in 2002, the M.S. degree in distributed computing from the University of Paris XI, Orsay, France, in 2003, and the Ph.D. degree in control and computer engineering from the University of Evry Val d'Essonne, France, in 2006. He is currently with the Technology, Computer Science and Applied Mathematics Division of Institut Français du Pétrole (IFP). His research interests include real-time and multi-domain simulation, control and process monitoring.

1 Introduction

Modeling and simulation technology is widely used within the automotive industry, where it has proven, in many projects, that it allows reducing the time-to-market. The support of simulation in the automotive design process is essential to fulfill costumers' needs or to comply with the new regulations. It provides the ability to predict the behavior and consequences of design choices, to access non measurable quantities or to perform test procedures which could be expensive or hazardous in real experimentations.

Modeling and simulation is a very wide area, which is used by different engineering fields at different stages of the product life-cycle. It encompasses various modeling formalisms [1] (ordinary differential equations (ODE), differential algebraic equations (DAE), partial differential equations (PDE), state charts, Petri nets...), abstraction levels (component level, system level) or physics fields (hydraulics, thermodynamics, mechanics, electronics, control, computer science...). This diversity naturally leads to the emergence of various modeling languages and tools.

The most precise models of the involved physical phenomena are generally given by 3D partial differential equations (Navier-Stokes, Maxwell, heat and wave equations for example). This precise modeling might be needed at specific design stages of some vehicle components. For example, designing new engines implementing innovative combustion concepts requires the study of the turbulent combustion and the chemical kinetics within the combustion chamber, with timescales in the order of the turbulent timescale [2, 3]. With this fine modeling, simulating one second might require several minutes or even hours on high performance computers.

In opposition to this fine granularity simulation, system simulation focuses on a higher-level view of the vehicle. It allows the study of a global performance, such as fuel consumption or road handling. It is also needed in the design, development and validation of control strategies [2]. The system simulation of a vehicle may be mainly characterized by the abstraction level (system level), the modeling formalisms (ODE, DAE, state charts...). However, it requires a multi physics approach, at different timescales. Simulation times on standard desktop computers may range from the real-time to some tens times the real-time.

The design, validation and performance assessment of new vehicle concepts, such as hybrid vehicles, are increasingly relying on system simulation. New simulation models (batteries, electric motors...) have to be included in order to allow the hybrid powertrain simulation. However, the exchange, reuse and exploitation of models are still limited. Boosting model exchange requires fulfilling expressed and non expressed needs of both *model suppliers* and *model users*. For model suppliers, the main needs are, on one hand, the ability to easily protect the confidential know-how contained in the model, and on the other hand, the ability to expose the model at the right abstraction level. But model users need the ability to quickly use the received models with a reasonable cost and effort. This requires the ability to understand the models without having to explore all their programming details and without having to be proficient in their modeling languages or development environments. In both situations, simulation speed performance is the key for testing more concepts in less time. The exploitation of multi-core processors for system simulation, which is currently lacking in major system simulation environments, is the major potential for such performance speedup.

This paper focuses on these new requirements. It introduces the xMOD platform, which have been developed at IFP. It illustrates the way xMOD addresses these requirements, its contributions to current design process, its simulation speed performance and presents its application to a typical use case characterizing the increase of complexity and the multi-domain interaction: the hybrid vehicle.

This paper is organized as follows. First, an overview of current modeling and simulation tools and methodologies limitations, within the automotive industry, is presented. Then, the xMOD platform concepts and contributions are presented. Finally, the application of xMOD to a hybrid powertrain design and validation is described.

2 Model exchange and exploitation barriers

Building a system model is usually performed through the assembly of its components' models. A hybrid vehicle is a typical example of a complex system. Its design, development and validation is performed in concurrent cycles, involving various teams, working on a wide range of fields including mechanics, thermodynamics, hydraulics, power electronics, heat transfer, vibrations, control... During these cycles, specific modeling and simulation tools might be preferred by these different teams, for example, AMESim or GT-POWER for engine modeling, Matlab/Simulink for control design, ASCET for control implementation, Dymola for vehicle dynamics, Comsol for Batteries modeling... Model development is an iterative process. Starting from a first version, models are progressively improved, refined, calibrated and validated. For those reasons, within one big entity, different models are available, at different abstraction levels and precision. However, the integration and exploitation of these models is still limited, mainly due to model exchange obstacles, and the lack of dedicated tools allowing model exploitation, and which separates model building from model exploitation.

2.1 Model exchange obstacles

In current model-based design processes, various obstacles limit the possibilities of model exchange. Among these obstacles, we may cite:

- The difficulty to integrate and exploit heterogeneous models without having to explore their pro-

gramming details and without having to be proficient in their modeling languages or development environments.

- Model exchange between different entities (for example, an OEM and a tier 1 supplier) requires the protection of the confidential know-how contained in the exchanged models. A current practice within the automotive industry is to exchange controllers or models as Matlab/Simulink mexw32 s-functions. Although this means allows to protect model equations, it does not allow to efficiently personalize the model interface (for example, selecting the variables or signals that may be monitored), nor to numerically integrate the s-functions with different solvers.

- Bridges (i.e. co simulation interfaces) between the different modeling and simulation tools do not always exist. Some existing bridges have inherent limitations, which reduce the co-simulation performance and possibilities [4].

- The lack of collaborative tools, ensuring the collaborative development and exchange of models during the whole development cycle.

2.2 Model exploitation handicaps

Moreover, there is a lack of tools allowing the efficient exploitation of models through off-line simulation. In fact, modeling and simulation environments are mainly targeted to models building. However, test, experimentation and validation tools, such as dSPACE' ControlDesk or National Instruments' LabVIEW provide virtual instrumentation facilities allowing simple, efficient and intuitive interaction with physical equipment. Virtual instruments like gauges, buttons, potentiometers, bar graphs or graphs allow the construction of user specified dashboards to interact and to monitor a physical system. However, this virtual instrumentation is mainly targeted to be interfaced with a specific hardware, and to be executed with hard real-time timing requirements. We believe that using this virtual instrumentation to interact with a simulated model, without having these hardware and real-time constraints, would provide an efficient way for model exploitation. In fact, virtual dashboards allow abstracting the modeling language in which a model has been construed. They may be customized in order to only display the features that a model user, in other department, division or company needs (abstraction capabilities).

2.3 Reference platform requirements

Based on these ideas, a set of requirements defining an ideal model exchange, integration and virtual prototyping platform have been formalized. This platform has to fulfill the following requirements:

- Facilitate the exchange of models between the different engineering teams.

- The exchange has to be based on a domain specific language (a minimal and simple language that captures the essential characteristics of all used modeling and simulation environments).

- Improve simulation speed by enabling the efficient exploitation of multi-core processors, and allowing the application of a different step-size and solver, to the each integrated model (depending on its stiffness).

- Be compatible with modeling and simulation standards.

- Be simple and user-friendly (intuitive).

These needs were the motivation behind the development of xMOD. In the following, the main concepts of xMOD are overviewed, and the way this platform addresses the aforementioned needs is illustrated.

3 The xMOD platform

3.1 Main concept

The xMOD concept relies on separating the phases of *model building* and *model exploitation* of the model lifecycle. It focuses on the latter phase. The main idea of xMOD is to combine, within the same platform:

- A heterogeneous model integration environment

- A virtual experimentation laboratory

xMOD aims to use the concept of "the gray to black" box as a main basis for model exchange. The exchanged models are instrumented using virtual instruments dashboards, allowing to abstract their modeling language. Figure 1 illustrates the concept of instrumented heterogeneous model representing the model exchange format that xMOD aims to promote.

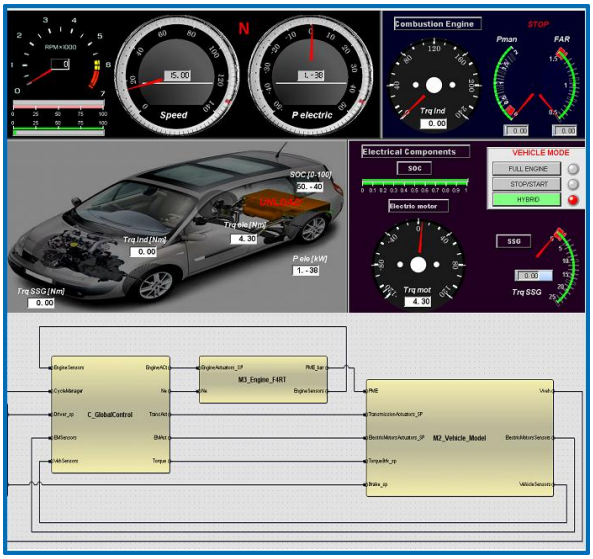


Fig. 1 Instrumented heterogeneous model in xMOD, representing the basis for model exchange

xMOD does not intend to replace the original modeling and simulation tools, but aims at promoting their coexistence. In fact, models lifecycle may be decomposed in two parts : model construction and model exploitation. xMOD targets to address this second phase of model exploitation.

3.2 Model import in xMOD

This feature is achieved by the *xMOD Target*, which provides a set of mechanisms allowing to convert a model to a unified representation which is independent from its original modeling environment. In xMOD, models are seen as gray boxes, whose "darkness" is customizable by the model owner. In this representation, models are characterized by inputs, outputs, parameters and signals.

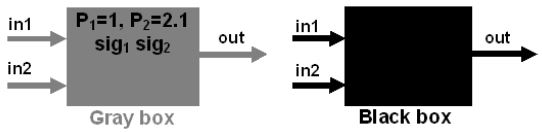


Fig. 2 Gray box vs. black box model

The xMOD target relies on Matlab Real-Time Workshop code generator. Exported parameters and signals may be selected by the model developer, prior to exporting its model to xMOD. This allows the model developer to protect its know-how. It may chose to hide all the parameters and signals of its model, and to export it as a black box. Other information (version, step-size...) is included in the model interface description xml file. Is worth mentioning that xMOD import format is open (and may be characterized by a xml document describing model interface and parameters values, and a C-code API). For that reason, users are also able to create own targets for their own simulation tools, which are not currently covered by xMOD.

3.3 xMOD contributions

The different case-studies that have been undertaken with xMOD show that the xMOD concept provides significant improvement to the current model-based design methodologies. It allows fulfilling expressed and non expressed needs of both model suppliers and model users. xMOD contributions may be summarized in the following points:

- **Heterogeneous model integration:** xMOD provides a heterogeneous model integration environment for models built by different persons using different languages and tools and working within different entities. Currently, xMOD allows the integration of Matlab/Simulink, LMS Imagine.Lab AMESim and Dymola models.
- **Virtual experimentation features:** xMOD allows creating virtual dashboards, containing virtual instruments, which may are linked to the different parameters or signals of the model that have to be displayed (and that the model designer allowed to make accessible). xMOD allows also to create automated virtual test benches using different scripting languages. These features enable validating the system model which was built through the integration of different components, dimensioning components (design parameters optimization using simulation), pre-calibrating control algorithms parameters or running robustness tests.
- **Simulation speed:** xMOD runs binary models compiled from their associated C code, activating the code full optimization during compilation. For that reason, execution times are several times faster than those obtained while simulating interpreted modeling languages. xMOD associates an execution thread to each integrated model. Mechanisms for guaranteeing the synchronization between the different threads were implemented. This allows xMOD to execute models embedding different solvers at different step-times on multi-core and multiprocessor architectures.
- **Standalone execution :** xMOD offers an integration, co-simulation and virtual prototyping platform, which is able to *run* models without any need to the existing modeling and simulation tools. xMOD allows the integration and exploitation of models without requiring the installation, on the same computer or network, of the original modeling tools (Simulink, AMESim or Dymola) that have been used to produce these models (in opposition to other collaborative co-simulation environments, which are based on tools coupling, and which ensures the co-simulation by running the models on their original simulation environments and exchanging data using DDE or DCOM protocols). Imported models embed all the needed data: no data files are referred to. This significantly eases model exchange and increases simulation performance.
- **Congregates the benefits of each tool:** xMOD allows to exploit the forces of each modeling and simulation tool (modeling language or environment suitability with an engineering field, solvers efficiency, language expressivity, modeler proficiency, existence of adapted libraries...).
- **Co-simulation benefits:** In comparison to a global simulation of a complete system using a single solver, a co-simulation where the system is conveniently decomposed, for example to isolate stiff parts, provides a more efficient execution and better simulation times [5]. In fact, isolating stiff components of the system model avoids constraining the whole model with their solving requirements. Non-stiff parts may be integrated with lighter solvers, and with greater step-sizes.
- **Confidentiality management and know-how protection:** xMOD executes models in binary form. When a model is converted to xMOD import format, two files are generated: a model interface

description xml file (whose content is customizable) and a dynamic link library, which is compiled directly from C code. This provides a generally acceptable level of model details protection. The xMOD target allows specifying exactly the signals and the parameters that have be included in the model interface and made accessible during model exploitation and co-simulation.

- **Life-cycle continuity:** Dashboards, model description format and automation scripts used in xMOD are compatible with HIL and test bench automation software in use in IFP (Morphee 2 from D2T). Thanks to this compatibility, the model-in-the loop validation performed with xMOD allows to prepare and to significantly reduce the time of the subsequent phases (which are more expansive).

4 A case study: Hybrid powertrain design and validation

xMOD has been used in the cooperative research project HyHIL, which involved IFP, D2T, LMS Imagine, G2Elab and Renault. This project focused on hardware-in-the-loop (HIL) applications to hybrid powertrains design and assessment. The main goal of this study was the evaluation of hybrid propulsion concepts and the benefits of different degrees of hybridization in a flexible architecture, by using a chain of simulation platforms: from the co-simulation to the high-dynamics engine-in-the-loop test bed, through a virtual version of the latter. In this project, xMOD has been used for the development and validation of the engine and vehicle simulation models, the associated controllers and the energy management supervisor (EMS).

4.1 Description of the analyzed parallel hybrid architecture

The parallel hybrid architecture analyzed in this case study is depicted in Figure 3. The engine is a gasoline turbocharged engine. The pre-transmission electric machine is a starter-generator (SSG), which is only allowed to start the engine, without any boosting or regenerating capabilities. In contrast, the post-transmission electric machine (MOT) allows for power assist, including purely electric drive, and battery recharge, including regenerative braking. The transmission ratio between the electric machine and the wheels is constant, while the gearbox is an automated manual transmission.

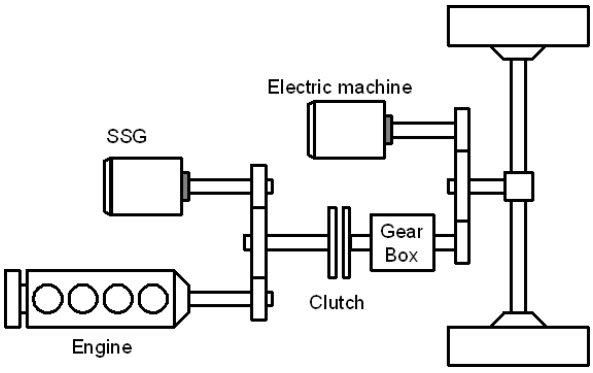


Fig. 3 The parallel hybrid architecture

4.2 Overview of hybrid vehicle models and controllers

After choosing the parameters of each vehicle component and computing the performance potentials, the next development stage of a hybrid vehicle is the control design. This phase generally relies on a simulator that integrates vehicle and powertrain models with the different controllers and supervisors. For an accurate design and validation of the controllers, it is necessary to use dynamic models.

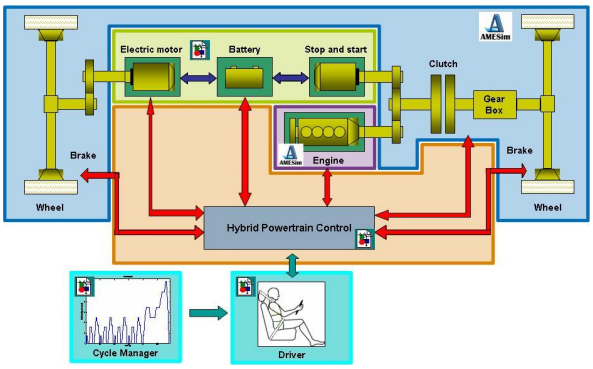


Fig. 4 Co-simulation architecture describing the involved models

Figure 4 illustrates the architecture of the hybrid vehicle simulator. The simulator includes the EMS detailed in [6]. The EMS and the low-level controllers were written in Matlab/Simulink. The controllers have been coupled with models aimed at representing the system dynamics in a much more accurate way than the simple models used to design and derive the optimization laws. The simulator is made of the following sub-models:

- A high-frequency zero-dimensional spark-ignition engine simulator developed in AMESim. It represents the main physical phenomena with a temporal resolution in the order of one crank angle. Such a precision is necessary to represent the impact, on the system, of the engine transient maneuvers, including start-ups. This model has an order of 115. Its maximum step-size is 100 μs (with any fixed-step solver).
- A high-frequency transmission model of order 14 (dual-mass flywheel, clutch and gearbox) developed in AMESim. The model stiffness required using a fourth-order Runge-Kutta solver.
- A battery model written in Simulink. In the presented version of this case study, an equivalent-circuit model of the battery was included. How-

ever, in the subsequent versions, electrochemical models of the battery will be used [7]. The xMOD integration facilities allows easily replacing the equivalent-circuit battery model with the electrochemical one.

- The cycle manager, allowing to apply a selected standardized cycle.
- The driver model, which simulates a human driver that tries to follow the specified cycle.
- The global controller, written in Simulink, and which includes the vehicle energy management system (EMS), the engine controller and low-level controllers (such as transmission controller). The EMS distributes the driver torque request taking into account optimization considerations. The engine and motor controllers try to apply the torque request from the EMS to the physical components. The EMS and the other controllers have specific execution rates ($1\ ms$ for the EMS, $1\ ms$ for the transmission controller, whereas the engine controller is asynchronous and driven on TDC (Top Dead Center)). The different involved sub-rates are handled using Simulink Triggering functionalities. Since interrupts are not available on a pure simulation environment on desktop PC, the global controller is executed as a discrete-time system, with a base period of $100\ \mu s$ (in order to allow a precise emulation of the TDC and crank angle-based sensors).

4.3 Integration and validation using xMOD

Building the full hybrid vehicle simulator requires integrating these Simulink and AMESim models. This integration was performed using the xMOD integration functionalities. A screen shot of the hybrid vehicle simulator integration in xMOD is given in Figure 5. Based on this integration project, an xMOD simulation

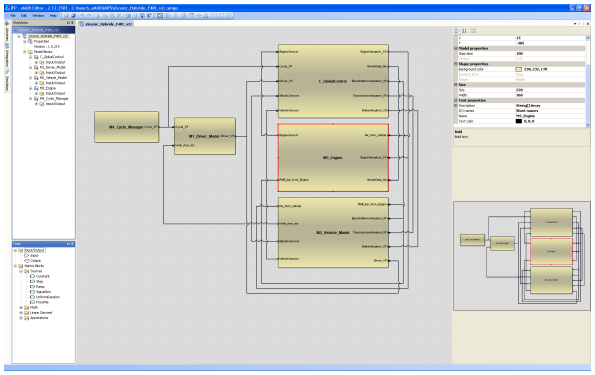


Fig. 5 Integration project of the hybrid vehicle in xMOD

project was constructed. It allows to link the different quantities (inputs, outputs, parameters and signals) of the hybrid vehicle model integration to virtual dashboards, as shown in Figure 6. These virtual dashboards were constructed based on a predefined instruments library, containing the most commonly used virtual instruments (such as graphs, gauges, buttons, bar-graphs, potentiometers and bitmap containers). Furthermore, xMOD allows the user to write scripts allowing to animate these virtual dashboards.

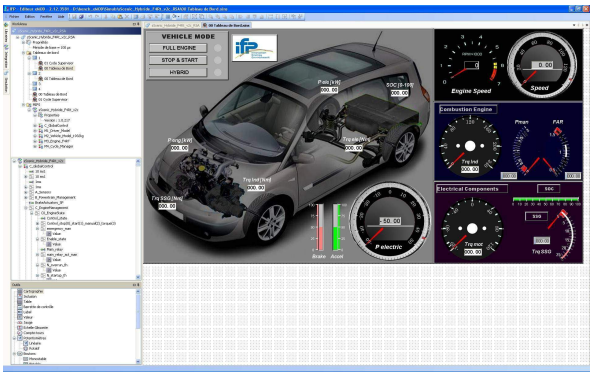


Fig. 6 Simulation project of the hybrid vehicle in xMOD

This xMOD simulation project was then used to validate the EMS and the low-level controllers on different driving tests (tip-in, tip-out, driving cycle) and to assess the behavior of the physical components models. It allowed to visualize in a simple and intuitive way the different choices of the EMS (starting or stopping the spark-ignition engine, starting and stopping the electric motor, charging the batteries...) during the driving cycle, and to observe their impact on fuel consumption.

Figure 7 shows a simulation dashboard under execution in xMOD, where the hybrid vehicle components (engine, electric motor, battery) are highlighted when active. In the particular screen shot of Figure 7, the electric motor is highlighted, which indicates that it is delivering the torque (whereas the spark-ignition engine is stopped).

4.4 Comparison between Simulink, xMOD in terms of simulation CPU time

Two integration setups of the simulator were tested and compared. First, the Simulink and AMESim models were integrated in Simulink (where the AMESim models were imported as s-functions). The integration of the hybrid vehicle models and controllers in Simulink imposes to apply *the same time-step* and *the same solver* to all the continuous-time models, especially the involved physical models. A fourth-order Runge-Kutta solver with a step-time of $100\ \mu s$ was selected in the Simulink integration.

In contrast, in xMOD integration project, each model may be *imported with its specific solver and time-step*. Each model is executed in a separate thread and may be potentially dispatched on a different processor core. For that reason, the AMESim vehicle model that was imported in xMOD was generated with a fourth-order Runge-Kutta solver and a solver step-time of $500\ \mu s$,

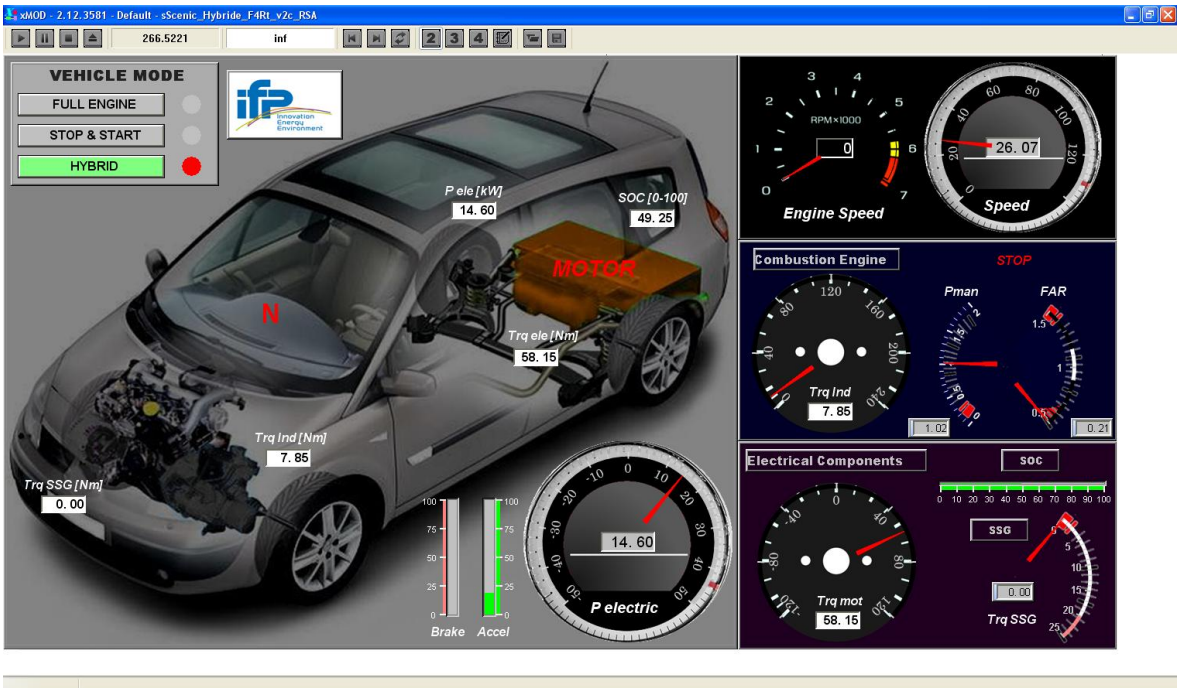


Fig. 7 Simulation execution in xMOD: EMS operation, torques and energies visualization

whereas the spark-ignition engine model was generated with a first-order Euler solver and a solver step-time of 100 μs. Cycle manager, driver and global control (including the EMS and all the low-level controllers) models are executed as discrete-time blocks (with no embedded solver) at the respective sample times of 10 ms, 10 ms and 100 μs. This co-simulation scheme led to a significant improvement of the simulation speed with respect to the Simulink-based integration, as shown in Table 1. This Table summarizes the CPU time needed to simulate, on a standard laptop PC with an Intel Core 2 Duo Processor at 2.40 GHz, the 1180 seconds of a NEDC cycle with both xMOD and Matlab/Simulink R2007b (which does not support yet multi-core simulation). Simulation results show that xMOD allowed to speedup the simulations with a factor of 6. This illustrates simulation performance potentials of xMOD (optimized C code execution and multithreaded execution on multi-core processors).

Tab. 1 Comparison of simulation CPU time (in sec) obtained with Simulink and xMOD

	Conventional Vehicle	Stop & Start	Hybrid
Simulink	7920	8025	9110
xMOD	1272	1284	1372
Speedup	**6.2**	**6.2**	**6.6**

4.5 Comparison between Simulink, xMOD and experimental results in terms of fuel consumption

Finally, the simulation results obtained with xMOD were compared to the Simulink simulator and to the experimental engine-in-the-loop test bench (where the engine is real but the vehicle is simulated in real-time). Table 2 compares the conventional engine-driven operation, the same operation with a stop-and-start capability suppressing the idle consumption, and the hybrid operation with the EMS (all under the NEDC drive cycle). The benefit in fuel consumption reduction using the hybrid capability and its energy management is clearly observed, particularly in urban driving conditions. Table 2 summaries also the fuel consumption results obtained using the Simulink simulator, the xMOD simulator and on the experimental test bench. The xMOD simulator results were closest to the experimental results. This is due to the multithreaded model execution semantics of xMOD, which are similar to the real-time model execution semantics of the test bench.

Tab. 2 Comparison of the different fuel consumptions (in $[l/100km]$) obtained with Simulink, xMOD and experimentally

	Conventional Vehicle	Stop & Start	Hybrid
Simulink	7.01	6.46	4.20
xMOD	7.25	6.69	4.42
Test bench	7.20	6.70	4.64

5 Conclusion

This paper introduced the new model integration and virtual experimentation xMOD platform, aiming at simplifying model exchange and exploitation. After reviewing the main motivations behind the development of xMOD, it described the main concepts and contributions of this software platform. The application of xMOD to the design and validation of a hybrid vehicle was then presented, allowing an easy visualization, interaction and understanding of simulation results, and an increased simulation performance with respect to the usual integration platform.

6 References

[1] P. J. Mosterman and H. Vangheluwe. Computer automated multi-paradigm modeling: an introduction. *Simulation*, 80(9):433–450, 2004.

[2] A. Albrecht, O. Grondin, F. Le Berr, and G. Le Solliec. Towards a stronger simulation support for engine control design: a methodological point of view. *Oil & Gas Science and Technology*, 62(4):437–456, 2007.

[3] O. Colin, A. Pires da Cruz, and S. Jay. Detailed chemistry based auto-ignition model including low temperature phenomena applied to 3d engine calculations. In *Proc. 30th International Symp. on Combustion, Pittsburgh*, pages 2649–2656, 2005.

[4] E. A. Lee and H. Zheng. Leveraging synchronous language principles for heterogeneous modeling and design of embedded systems. In *Proc. International Conf. on Embedded Software*, Salzburg, Austria, 2007.

[5] J. Larsson, P. Krus, and J-O. Palmberg. *Power Transmission and Motion Control*, chapter Methods for Organising Co-simulation Among Several Participants. Professional Engineering Publishing Limited, London and Bury St Edmunds, UK, 1999.

[6] A. Chasse, Gh. Hafidi, Ph. Pognant-Gros, and A. Sciarretta. Supervisory control of hybrid powertrains: an experimental benchmark of offline optimization and online energy management. In *Proc. 2009 IFAC Workshop on Engine and Powertrain Control, Simulation and Modeling, Rueil-Malmaison*, 2009.

[7] V. Sauvant-Moynot, E. Prada, J. Bernard, J. Martin, A. Sciarretta, N. Rajapakse, Y. Touzani, J.-C. Dabadie, and F. Badin. An integrated approach to high power modeling: from the electrochemistry to the vehicle. In *Proc. International Battery, Hybrid and Fuel Cell Electric Vehicle Symp. & Exhibition, Norway*, 2009.