Implementation of an Extended Vehicle Model Architecture in Modelica for Hybrid Vehicle Modeling: Development and Applications

John Batteh Michael Tiller
Emmeskay, Inc.
Plymouth, Michigan USA
jbatteh@emmeskay.com mtiller@emmeskay.com

Abstract

This paper outlines the development and implementation of a vehicle model architecture for hybrid vehicle modeling. The architecture is based on the VehicleInterfaces library with significant extensions to enable more flexible, configurable implementations for hybrid vehicle applications. Additional elements are added to the interfaces and architecture to allow more flexible electrical system modeling and more detailed thermal modeling. Four different hybrid vehicles are implemented as sample applications using the newly-developed architecture. The scheme and canonical library structure for the component, subsystem, and system models is also discussed to document a mechanism for user-friendly handling of parameterized models and fully-implemented models in a complex model architecture with extensive model data. Models and simulation results are shown for the Toyota Prius, Lexus RX400h, a concept hybrid sedan, and a concept hybrid sport utility vehicle (SUV). Extensions to VehicleInterfaces are also proposed to enhance the library to include additional features to improve support for future conventional and hybrid vehicle modeling efforts.

Keywords: hybrid vehicles; vehicle modeling; model architecture; VehicleInterfaces

1 Introduction

Since the introduction of the Toyota Prius in the U.S. in 2000, hybrid vehicles have been gradually gaining acceptance in the U.S. as more consumers become aware of fuel economy and the effect of atmospheric CO_2 on climate change. While existing tax credits and government incentives have provided some stimulus for hybrid vehicle purchases, the overall share of hybrid vehicles in the light duty segment is still less than 2% as shown in Figure 1. However, hybrid vehicle share is expected to increase substantially over the next 10 years as more manufacturers introduce hybridized vehicles. The share of hybrid vehicles is projected to reach nearly 9% in 2015 as shown in Figure 1.

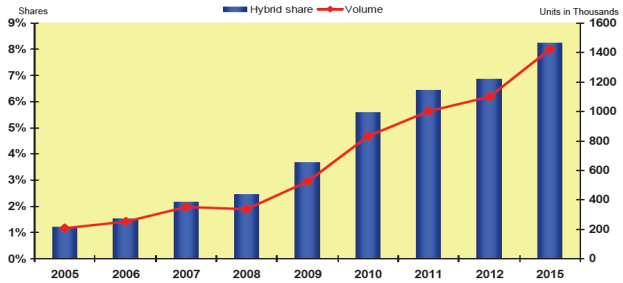


Figure 1. Projected US hybrid vehicle sales

Given the accelerated introduction of hybrid vehicle models over the next several years, there is an increasing need to develop analytic tools to reduce development time for these vehicles which are significantly more complex than conventional vehicles. These analytic tools can be used to assess the impact of different hybrid architectures, size/design the components, perform tradeoff and robustness studies, provide component specifications based on vehicle targets, and develop/optimize the control strategy and subsequent calibration to balance vehicle attributes.

Modelica has been used extensively for vehicle system modeling [2]-[6]. With a growing list of commercial, free, and internally-developed OEM proprietary model libraries, the need for a unifying vehicle model architecture was quickly realized. The purpose of a standardized model architecture is to provide consistent interfaces and system decomposition to promote plug-n-play interoperability between libraries. The first vehicle modeling architecture in Modelica was VMA [7]. Released in 2003, VMA was based on a Ford-internal architecture. After additional feedback from library vendors and end users, VMA was subsequently modified and released as the VehicleInterfaces library in 2006 [8]. The objective of VehicleInterfaces is to provide an open architec-

DOI: 10.3384/ecp09430109

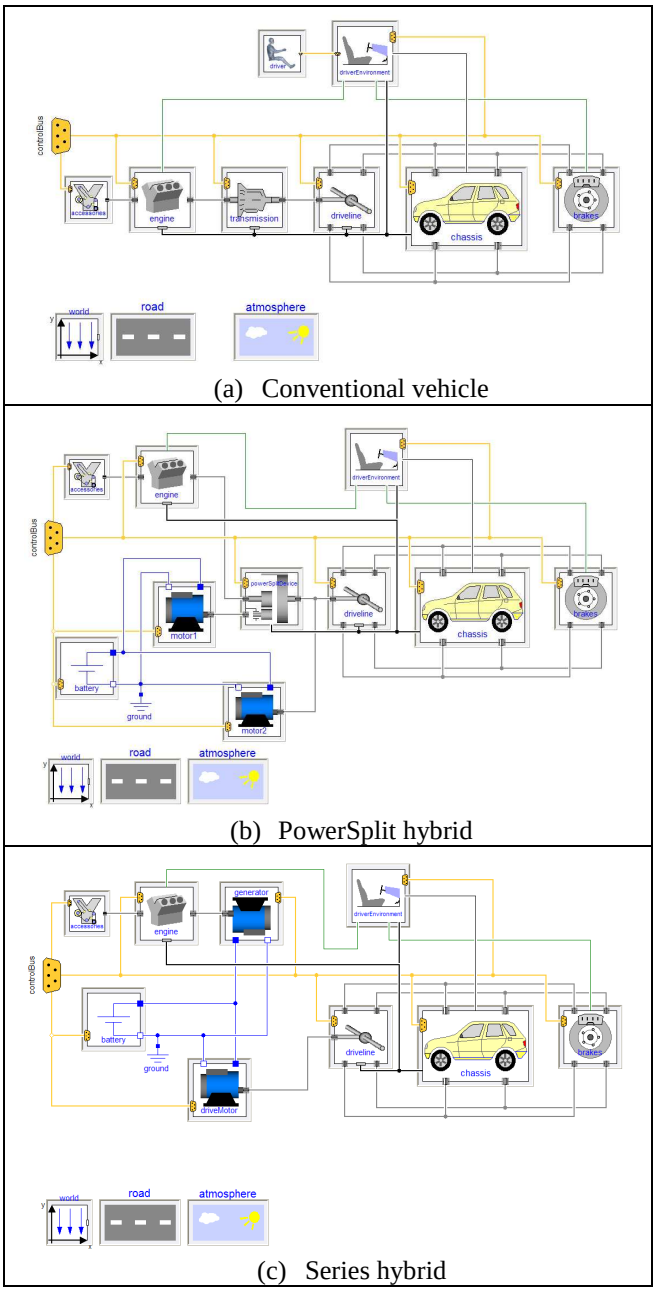
ture to support configurable modeling of both conventional and hybrid vehicles. The library has been used as the starting point for several vehicle modeling applications [6] and is still under development.

This paper outlines the development and implementation of an extended vehicle model architecture based on VehicleInterfaces with additional enhancements to better support hybrid vehicle modeling. Extensions have been made to the interfaces and additional components added to the architecture to enable more flexible, configurable implementations for hybrid vehicle applications. Four different hybrid vehicles, namely the Toyota Prius, Lexus RX400h, a concept hybrid sedan and SUV, are implemented as sample applications using the newly-developed architecture. Sample drive cycle simulations are shown for the four vehicles. The scheme and canonical library structure for the component, subsystem, and system models is also discussed to document a mechanism for user-friendly handling of parameterized models and fully-implemented models in a complex model architecture with extensive model data. Finally, extensions to VehicleInterfaces are proposed to enhance the library for future conventional and hybrid vehicle modeling efforts.

2 Architecture Development

2.1 VehicleInterfaces Examples

The VehicleInterfaces library includes example model architectures for many different types of vehicles, including conventional and hybrid vehicles. Example architectures from VehicleInterfaces 1.1 are shown in Figure 2 for a conventional (a), PowerSplit hybrid (b), and series hybrid (c) vehicle.



(a) Conventional vehicle

(b) PowerSplit hybrid

(c) Series hybrid

Figure 2. Example architectures for conventional and hybrid vehicles from the VehicleInterfaces library

While the conventional vehicle architecture seems quite suitable, the two hybrid vehicle architectures do not appear to offer a similar system decomposition to enable modeling flexibility at the system level. In particular, these example hybrid architectures do not appear to implement a formal electrical subsystem nor are the elements of the hybrid drivetrain grouped at the subsystem level. These features are required to support plug-n-play modeling at the system level with model components of varying level of detail. It should be noted that the hybrid vehicle architectures are appropriate for some model implementations but simply may not provide enough flexibility for models of varying level of detail with minimal changes to the top-level architecture.

2.2 New Architecture

Given the observations noted in the previous section regarding the example architectures in Vehicle Interfaces 1.1, a new architecture was developed based on the following design criteria:

- Extension from VehicleInterfaces design to maximize compatibility with existing model libraries
- Single model architecture that supports both conventional and hybrid vehicle models
- Additional support for electrical and thermal systems

To meet the design criteria above, the extended vehicle architecture shown in Figure 3 was developed. There are several interface models from the VehicleInterfaces library which required little or no modification. These models include the driver, world, road, and atmosphere components. The remaining interfaces are either modified or newly-added and will be discussed in detail next.

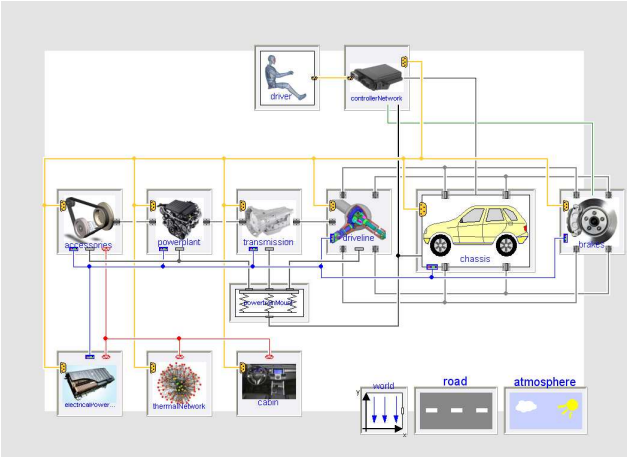


Figure 3. Model architecture

To support the proliferation of electrical components throughout modern vehicle subsystems, an electrical bus connector was added to the accessories, powerplant, transmission, driveline, chassis, and brakes subsystems. The electrical bus is an expandable connector that supports both single and multivoltage representations of the vehicle electrical system. Note that it is not required to terminate the electrical connection in component implementations which do not interact with the electrical system. As a result, no special provisions must be made for handling electrical connections in subsystem models that do not interact with the electrical bus.

In an effort to formalize the electrical subsystem, a new component is added for the electrical power network. The electrical power network is meant to represent the source of electrical power for the vehicle. Implementations of this subsystem could include a single battery, multiple batteries, power converters, and other components that provide and transform electrical power for use by the other subsystems.

A thermal bus was added to several components in the architecture. The thermal bus is also implemented as an expandable connector. The thermal bus was added to the electrical power network and accessory subsystems to facilitate modeling of the HVAC system for both vehicle and electrical system cooling. It should be noted that the thermal bus could also be added to the other vehicle subsystems to support thermal modeling of the engine, transmission, driveline, chassis, and brakes as shown in Section 6. A new cabin component was added to support thermal modeling of the cabin environment. A new thermal network component was added to provide the thermal linkages between the various interacting thermal components. These linkages could include cooling provided from the HVAC components in the accessories to the electrical power network and cabin components, thermal pathways between the electrical power network and the cabin, and thermal linkages between the vehicle and external environment. The addition of the thermal network component provides additional flexibility to modify the thermal routing between components without requiring modification of the models that implement the thermal capacitances.

The design of the electrical and thermal networks decouples the mechanical, electrical and thermal architectures. In this way, the electrical power and thermal network subsystem models allow complete different architectures for those subsystems to be implemented in a way that is orthogonal to the mechanical architecture.

With the ability to internally ground the reaction torques in the various component models in the Modelica Standard library, the impact of the various models on the powertrain mounts is often easily overlooked. Thus, powertrain mounts were also added to the vehicle architecture to encourage consideration of the impact of the drivetrain on the mounting system.

To support modeling of the vehicle control strategy, a controller network component was added to the vehicle architecture. The controller network can support both a single and distributed controller architecture as shown in the interfaces in Figure 4. Note the vehicle system controller which interacts with the driver interface and component controllers. Sample component controllers are engine, transmission, battery, driveline, climate control, motor, generator, *etc.* depending on the vehicle architecture. The functional form of these controllers is flexible enough that they can be mapped to hardware control units if desired. The controller network interface is flexible and configurable to allow the addition of other controllers, implementation of controllers of varying levels of detail, and controller implementations natively in Modelica along with external implementations such as C code and Simulink.

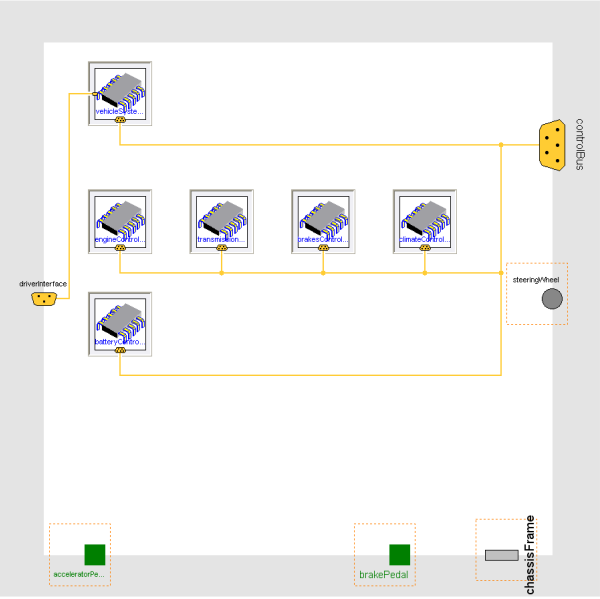


Figure 4. Distributed controller network interfaces

3 Canonical Library Structure

Despite the formal Modelica language features for model configuration, managing model variants and parameter data is a challenge in complex, hierarchical models. The challenge exists not only for the initial library developer but also subsequent model developers and end users. This section describes a canonical library structure implemented as part of the vehicle architecture and implementation effort. This structure was implemented in an effort to satisfy the needs of the model developer while balancing usability concerns for the end user. The guiding principles behind this structure are as follows:

- Promote object-oriented modeling of plant and controller subsystems by composition from reusable, parameterized components
- Provide a model package structure consistent with the model architecture and within which it is easy to find existing models and place new models
- Parameterize models at all levels (subsystem, component, and primitive) to promote model reuse
- Clearly separate generic, parameterized models from specific model implementations
- Implement a data model that preserves the integrity of parameter data throughout the model life cycle

The key design element of the canonical library structure is the separation and clear distinction between parameterized models and model implementations. Parameterized models include all relevant equations for simulation but do not specify any design parameter values. Model implementations extend from the parameterized models and provide the parameter design values. In this structure, explicit model implementations exist as named, fully-specified entities in the package hierarchy rather than *ad hoc* implementations created by specifying parameter values at instantiation. The advantages of named model implementations are as follows:

- No need for separate data package hierarchy as parameter data is specified directly in model implementations
- Implemented models clearly separated for model users
- Fully specified implementations consistently used in architecture, component tests, *etc.* without requiring any additional data to be provided by user
- Parameterization clearly identified as a task at creation of implementation model and not model instantiation
- Integrity of parameter data in model implementations can be maintained based on design choices initiated by model developer
- Implementations offer true plug-n-play capability in architecture without requiring subsequent modifications, thus integrating nicely with the replaceable concept in Modelica and tool implementations including multiple redeclares

The following figures show a sample implementation of the canonical library structure. Figure 5 shows the top level package structure which contains the interfaces package and the packages for the parameterized models. These packages can include additional subpackages to further classify the parameterized models. Note that these packages do not contain any implementations. Figure 6 shows the vehicle implementations package with implementations for the Prius and Lexus RX400h. An exploded view of the Prius implementation package is shown in Figure 7.

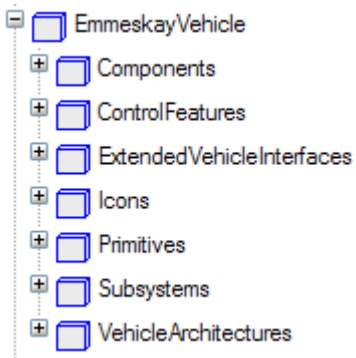


Figure 5. Top level package structure

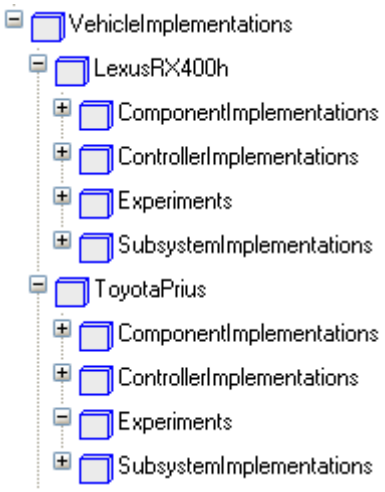


Figure 6. Vehicle implementations package structure for Toyota Prius and Lexus RX400h

Figure 7. Toyota Prius implementation

To support this library structure, two different types of parameterization are defined: parameterized model and configurable models. The characteristics of a parameterized model are defined as follows:

- Parameters declared in public section of Modelica model
- Can include instantiation of non-replaceable models
- Model can be instantiated
- Parameter values provided at instantiation and parameters can be propagated to higher level model
- Parameter values can be modified by higher level component
- Parameter values can be modified after compilation

The characteristics of a configurable model are as follows:

- Parameters declared in protected section of Modelica model
- Can include instantiation of replaceable components
- Model denoted as "partial" to indicate that it is not complete and can only be instantiated as a replaceable component in another model
- Explicit model implementations which are stored in the package hierarchy are required
- Model implementations are created by extending from the configurable model, providing parameter data, and selecting implementations for other configurable models
- Model implementations can be used directly in other models or tests
- Parameter values cannot be modified by higher level components at instantiation
- Parameter values can be modified after compilation

Ultimately, the type of parameterization used is defined by the model developer when the model is created. Some factors to be considered are the complexity of the parameter data, desired integrity of the parameter data, and the anticipated usage of the model. It should be noted that virtually all the models in the Modelica Standard Library are parameterized models according to the characteristics above. Figure 8 shows a sample configurable transmission subsystem model. This model is comprised of two replaceable configurable models for the `torque_converter` and `gearbox` components and one parameterized model for the inertia component.

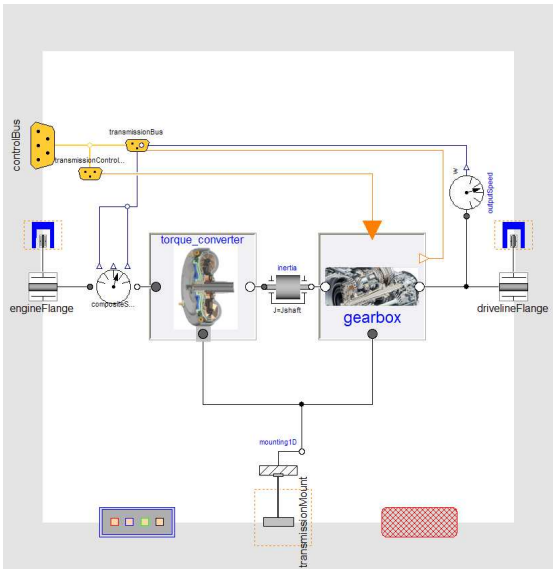


Figure 8. Sample configurable model

4 Hybrid Vehicle Implementations

Using the newly developed architecture, four sample vehicle implementations were created. The implemented models include the Toyota Prius, Lexus RX400h, and concept versions of a hybrid sedan and SUV. While the vehicle model architecture obviously supports models of varying level of detail and a wide range of engineering analyses, these implementations were focused on drive cycle simulations for fuel economy. The parameterization data for these models was collected from available publications in the open literature and from the last publicly-available version of ADVISOR [9].

The vehicle model implementations include the following subsystem representations:

- Accessories including performance-oriented model of vehicle air-conditioning system
- Mapped engine model
- Various implementations of conventional and hybrid transmissions with motors, gear seats, clutches, *etc.*
- Rigid front wheel drive (FWD) drivelines
- Vehicle chassis with lumped vehicle inertia, no-slip tires, and loads for aerodynamic drag and rolling resistance
- Simple brakes with prescribed actuation
- Dual voltage electrical power networks with fixed capacity battery models including battery thermal response
- Thermal networks including routing for battery and cabin cooling
- Lumped cabin models for vehicle cooling
- Controller network implementations including vehicle system, engine, transmission, battery, motor, generator, climate, and brake controllers
- Driver models based on drive cycles with capability to run both forward and backward models

The acausal nature of the Modelica modeling language enables several nice features of the model architecture:

- Ability to run both forward and backward drive cycle simulations with change only to the driver model (assuming underlying model is invertible)
- Ability to use model inversion to implement control features
- Ability to re-use physical, validated models across subsystems and applications
- Ability to plug-n-play models of varying level of detail to enable a wide range of engineering analyses to support model-based engineering over the entire product development process

4.1 Toyota Prius

The vehicle model implementation for the parallel hybrid Toyota Prius is shown in Figure 9. The implementation of the transmission subsystem for the PowerSplit transmission contains the motor, generator, and gearing components consistent with the hybrid transmission delineation in the Toyota drivetrain schematic [10] shown in Figure 10.

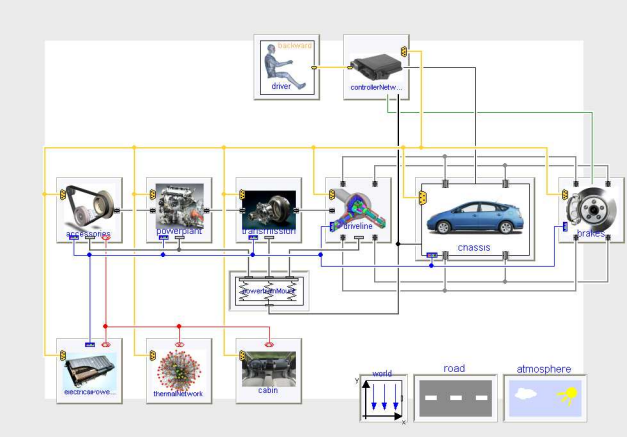


Figure 9. Toyota Prius model

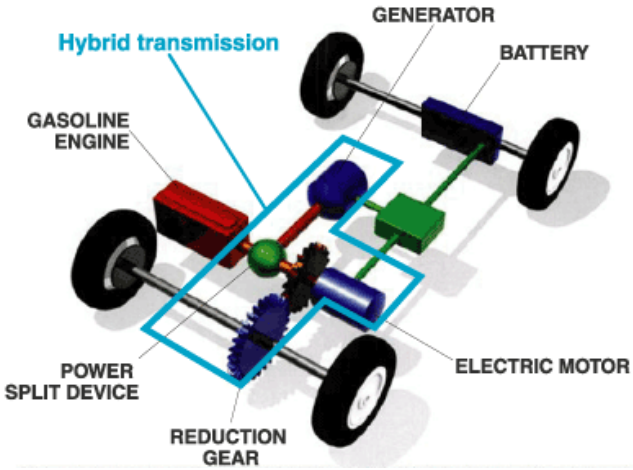


Figure 10. Toyota Prius drivetrain schematic [10]

4.2 Lexus RX400h

The vehicle model implementation for the Lexus RX400h is shown in Figure 11. Like the Toyota Prius, the Lexus RX400h is a parallel hybrid vehicle with a PowerSplit transmission. The drivetrain schematic in Figure 10 is applicable to the Lexus RX400h as well.

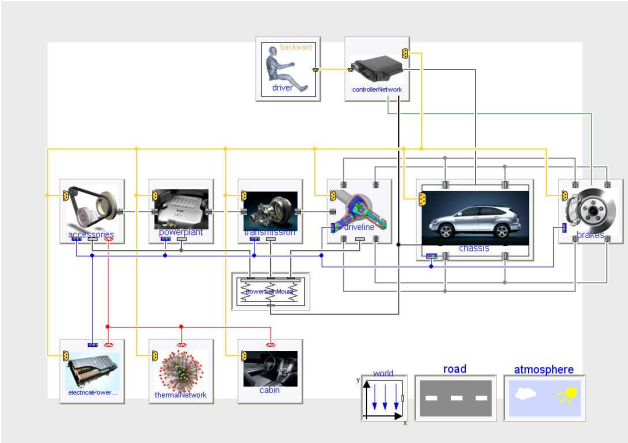


Figure 11. Lexus RX400h model

4.3 Concept Hybrid Sedan

The vehicle model implementation for a concept hybrid sedan with a parallel hybrid architecture is shown in Figure 12.

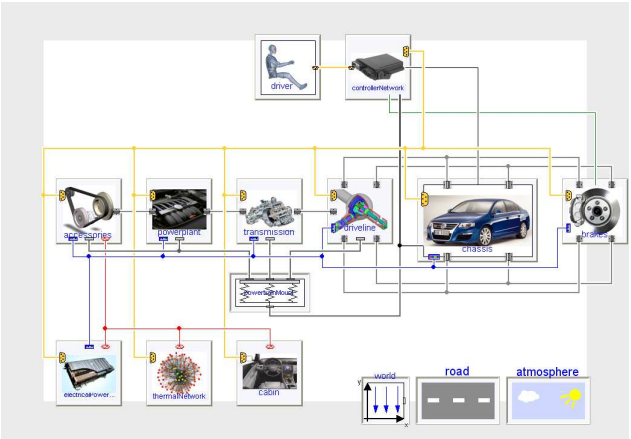


Figure 12. Concept hybrid sedan model

4.4 Concept Hybrid SUV

The vehicle model implementation for a concept hybrid SUV with a parallel architecture is shown in Figure 13.

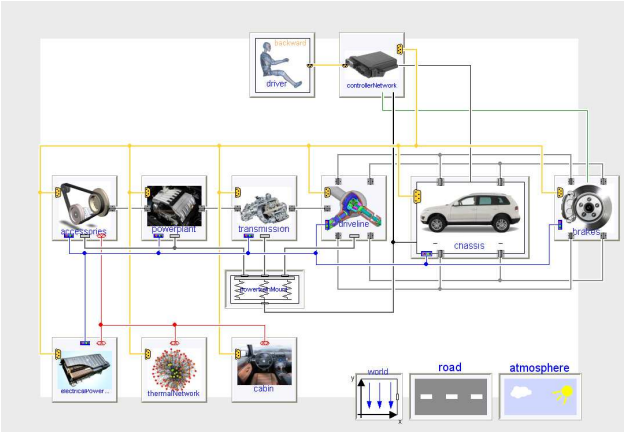
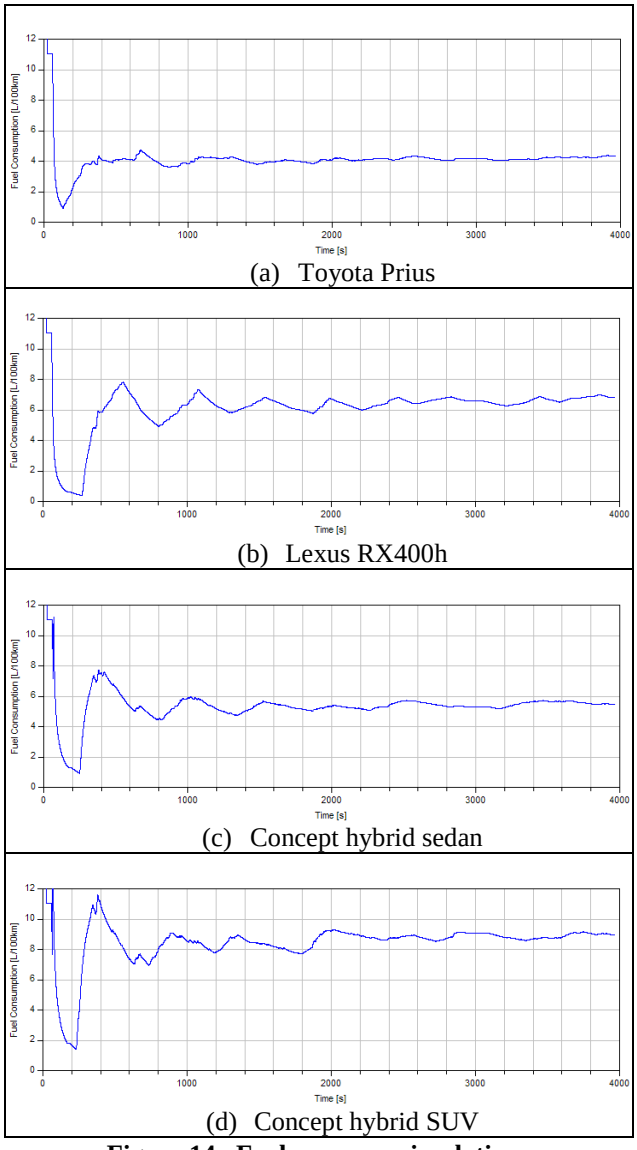


Figure 13. Concept hybrid SUV model

5 Drive Cycle Simulations

Sample drive cycle results from the four vehicle implementations are shown in this section. Fuel consumption data in L/100km is shown in Figure 14 for the four vehicles. The drive cycle is a proprietary cycle developed based on real-world driving over a range of conditions of interest to hybrid vehicle development.



(a) Toyota Prius

(b) Lexus RX400h

(c) Concept hybrid sedan

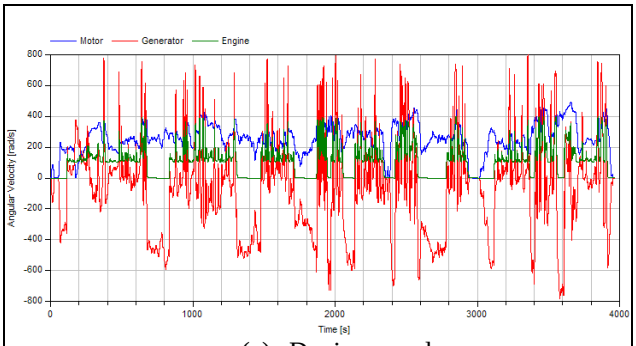
(d) Concept hybrid SUV

Figure 14. Fuel economy simulations

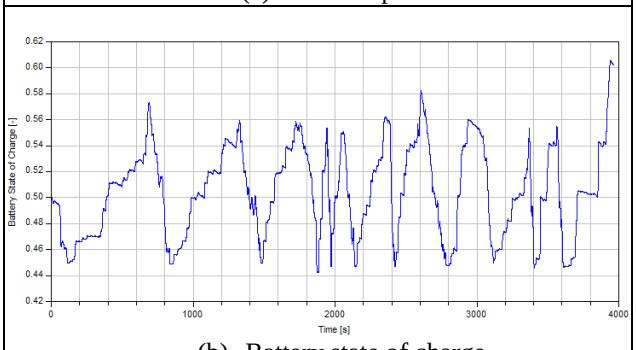
While every attempt was made to incorporate actual vehicle parameter data into the simulations, certain key parameters and component specifications were not available and thus were implemented based on the authors' best engineering judgment or based on appropriate scaling from existing data. In addition, drive cycle fuel consumption is highly dependent on the implementation and calibration of the vehicle control strategy. While control strategies were im-

plemented for all four vehicles, these strategies may not be representative of the actual, proprietary control strategies for the production vehicles. Thus, the fuel economy results should be viewed as representative only. Furthermore, it should be noted that there is no experimental data with which to compare the model as these vehicles either do not exist yet in hardware or were not actually driven over this drive cycle. However, the results appear reasonable and follow the expected trends.

Figure 15 shows some additional signals from the Prius drive cycle simulations. The top graph shows the speeds of the engine, motor, and generator during the drive cycle. The bottom graph shows the state of charge (SOC) in the high voltage battery. The resulting battery dynamics include the contributions of the vehicle system and battery control characteristics and charge/discharge due to driving requirements and regenerative braking.



(a) Device speeds



(b) Battery state of charge

Figure 15. Prius drive cycle results: device speeds and battery state of charge

6 Extensions to VehicleInterfaces

The VehicleInterfaces library [8] provides a solid architecture to support vehicle system modeling. The library offers substantial flexibility in modeling the mechanical (both 1D and 3D) and control system interactions in the vehicle. Based on the extensions to the library implemented as part of this work, this section proposes additions to VehicleInterfaces to enable improved support for future vehicle modeling efforts.

6.1 Electrical Modeling

Electrification of nearly all major vehicle subsystems in both conventional and hybrid vehicles necessitates system leveling modeling of electrical systems. Currently VehicleInterfaces does not include electrical connectors and interactions at the subsystem level. The following extensions to VehicleInterfaces would improve the library's ability to support vehicle modeling including electrical system effects:

- Addition of expandable electrical bus to all major vehicle physical subsystems as shown in Figure 16
- Addition of electrical power network subsystem to serve as architecture placeholder for electrical energy sources, converters, *etc.* which distribute electrical power via the expandable electrical bus to other vehicle subsystems

These extensions eliminate the need to extend the existing interfaces in VehicleInterfaces simply to add an electrical bus. In addition, the formal inclusion of an electrical system will natively allow modeling of hybrid vehicle architectures in a standardized architecture as shown in Figure 16 without having to add electrical components in an *ad hoc* way to the top-level architecture.

6.2 Thermal Modeling

System level thermal modeling is another key element of vehicle system modeling. Currently VehicleInterfaces does not include thermal interactions at the subsystem level. The extended vehicle model architecture shown in Figure 3 includes the addition of an expandable thermal bus to a few top level subsystem components. As mentioned in Section 2.2, the most flexible implementation would include the addition of the thermal bus to all major vehicle subsystems as shown in Figure 16. Including an expandable thermal bus eliminates the need to extend the existing interfaces in VehicleInterfaces simply to accommodate thermal modeling. The thermal network and cabin subsystems are an integral part of the thermal architecture for the simulations shown in Section 5 but could be omitted from a standard architecture in an effort to minimize top level subsystems.

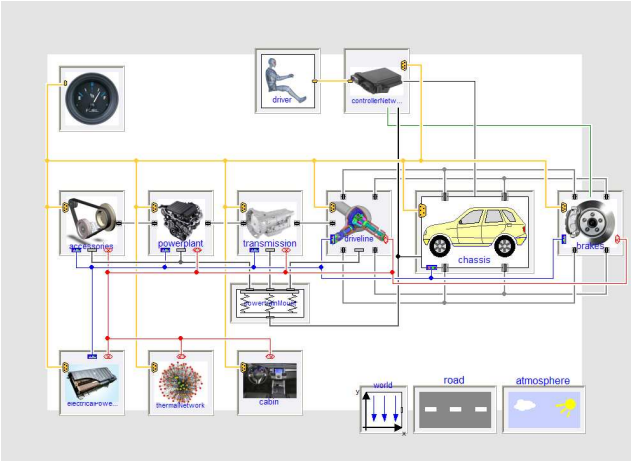


Figure 16. Sample extended architecture

7 Conclusions

This paper documents the development and implementation of an extended vehicle model architecture for hybrid vehicle modeling. This architecture is based on VehicleInterfaces and more easily enables flexible, configurable modeling of different hybrid vehicle configurations without the need for several different architectures. Additional elements have been added to the interfaces and architecture to allow more flexible electrical system modeling and more detailed thermal modeling. To illustrate the usage of this architecture, four different hybrid vehicles have been implemented and sample drive cycle simulations results shown. The canonical library structure implemented in this work has proven very capable of handling model development and implementation of model variants in a user-friendly way that integrates well with the formal model configuration language elements in Modelica. The canonical library structure has been discussed in detail along with a sample package implementation for the vehicle implementations shown in this work. Extensions to VehicleInterfaces have been proposed to improve the library for future vehicle modeling efforts.

Acknowledgements

The authors would like to acknowledge Hubertus Tummescheit and Magnus Gafvert from Modelon for initially proposing the scheme and canonical library structure implemented in this paper. Their contributions and insights were extremely valuable and are gratefully acknowledged.

References

[1] J.D. Power Automotive Forecasting, "US Hybrid-Electric Vehicle Sales Forecast Q3 2008", 2008.

[2] Tiller, M., Tobler, W.E., and Kuang, M., "Evaluating Engine Contributions to HEV Driveline Vibrations", Proceedings of 2nd International Modelica Conference, pp. 19-24, 2002. http://www.modelica.org/events/Conference2002/papers/p03_Tiller.pdf

[3] Laine, L. and Andreasson, J., "Modelling of Generic Hybrid Electric Vehicles", Proceedings of 3rd International Modelica Conference, pp. 87-94, 2003. http://www.modelica.org/events/Conference2003/papers/h26_Laine.pdf

[4] Hellgren, J., "Modelling of Hybrid Electric Vehicles in Modelica for Virtual Prototyping", Proceedings of 2nd International Modelica Conference, pp. 247-256, 2002. http://www.modelica.org/events/Conference2002/papers/p32_Hellgren.pdf

[5] Simic, D., Giuliani, H., Kral, C., Gragger, J., "Simulation of Hybrid Electric Vehicles", Proceedings of 5th International Modelica Conference, pp. 25-31, 2006. http://www.modelica.org/events/modelica2006/Proceedings/sessions/Session1b1.pdf

[6] Simic, D., and Bauml, T., "Implementation of Hybrid Electric Vehicles Using VehicleInterfaces and the SmartElectricDrives Libraries", Proceedings of the 6th International Modelica Conference, pp. 557-563, 2008. http://www.modelica.org/events/modelica2008/Proceedings/sessions/session5c.pdf

[7] Tiller, M., Bowles, P., and Dempsey, M., "Development of a Vehicle Modeling Architecture in Modelica", Proceedings of 3rd International Modelica Conference, pp. 75-86, 2003. http://www.modelica.org/events/Conference2003/papers/h32_vehicle_Tiller.pdf

[8] Dempsey, M., Gafvert, M., Harman, P., Kral, C., Otter, M., and Treffinger, P., "Coordinated Automotive Libraries for Vehicle System Models", Proceedings of 5th International Modelica Conference, pp. 33-41, 2006. http://www.modelica.org/events/modelica2006/Proceedings/sessions/Session1b2.pdf

[9] National Renewable Energy Laboratory (NREL), “Advisor documentation”, www.ctts.nrel.gov, 2002.

[10] Toyota Motor Corporation, “Toyota Hybrid System II: Hybrid Transmission”, 2009. http://www2.toyota.co.jp/en/tech/environment/ths2/hybrid.html

