High-Fidelity Transmission Simulation for Hardware-in-the-Loop Applications

Paul Goossens, Maplesoft; Orang Vahid, Maplesoft; Ang Zhao, Maplesoft; Takashi Iwagaya, Cybernet Systems Japan; Yoshihiko Nishi, Aisin AW; Yakayuki Kubo, Aisin AW

ABSTRACT

As automotive manufacturers strive to improve fuel-efficiency of their products, all components and sub-systems that go into a vehicle are under closer examination in order to minimize energy losses. In transmissions in particular, a great deal of effort is being put into assessing exactly how much power is lost, and what can be done to reduce those losses and improve overall efficiency. By investigating the losses at a very detailed level, new high-efficiency transmissions can be produced, through optimizing existing configurations, adopting new configurations, employing new innovations in components, and developing advanced controllers. These investigations are greatly enhanced through the use of physics-based simulation tools to create virtual prototypes of the transmission systems. These yield more efficient products at significantly reduced costs, by allowing engineers to address design issues long before investing in the physical prototype stage. These advantages are further enhanced if these high-fidelity models can be used to develop and test powertrain controllers in a real-time simulation environment.

This paper will present the use of symbolic technologies from Maplesoft for developing highly-efficient code for real-time implementation of transmission models with loss data included. It will also cover results from a recent Hardware-in-the-Loop testing project that demonstrate that the symbolic approach to model code-generation for real-time implementation can deliver dramatic execution speed improvements over other modeling approaches.

INTRODUCTION

As automotive manufacturers strive to meet and exceed performance requirements on fuel efficiency and ride comfort, they have focused increasingly on the transmission design as one of the key factors. Engineers are putting tremendous effort into determining exactly how the power is lost, and what can be done to reduce losses and improve overall fuel efficiency. As a result, the transmission industry is now actively involved in optimizing existing transmission designs and exploring new system architectures. At the same time, transmission controllers are becoming more complicated and more detailed product testing is needed than ever before.

Model-based development (MBD) plays a central part towards achieving these goals. Design iterations are done through virtual prototypes of the transmission systems, used in hardware-in-the-loop (HiL) simulations. Virtual prototyping can yield more efficient products at significantly reduced costs by allowing engineers to address design issues long before they invest in physical prototypes. Modeling and simulation environments like MapleSim™ have become critical tools in MBD and are being increasingly adopted by powertrain and transmission manufacturers.

In this paper we report on some of the activities under taken at AISIN AW in Japan regarding HiL simulation and the use of MapleSim environment to accelerate the development of automatic transmissions. The requirements for low calculation cost plant models for real-time simulations were met by creating the gear train part of the model in MapleSim. These models are then exported as optimized c-code for implementation into the HiL system.

The transmission models referred to in this paper are built using the components from the MapleSim Driveline Component Library (DCL) [1] as well as other components from the Standard Moldelica Libraries [2]. DCL covers all stages in a powertrain model from the engine through to the differential, wheels and road loads (See Figure 1).

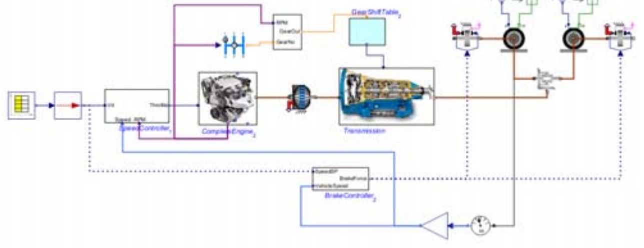


Figure 1. Full vehicle model in MapleSim

In Section 2, a brief overview of some of the components and features of DCL is presented. In Section 3, some aspects of the underlying symbolic technology that is utilized by MapleSim towards optimized code generation for real-time application are discussed. Section 4, is dedicated to some of the activities in AISIN AW in modeling automatic transmissions and HiL simulations. Conclusions are summarized in Section 5.

TRANSMISSION MODELING USING THE DRIVELINE COMPONENT LIBRARY

The MapleSim Driveline Component Library is a comprehensive collection of components and models that provide the fundamental components needed to build and simulate complex transmission models. The acausal model representation allows for rapid model development. The open structure of the components makes it very easy to modify them to suit specific requirements. Furthermore, the library allows for flexible inclusion of power loss data that best reflect the way in which the loss data was acquired. The DCL components are grouped into the following categories:

• Simple Gear Sets

• Compound Gear Sets

• Multi-speed Actuation Components

• Clutches and Brakes

• CVT and Torque Converter

• Engines and Dynamometers

• Loss Elements

• Differential Gears

• Vehicle and Tire

• Gear Selectors

In the remainder of this section, some of the features of the components used in modeling transmissions are discussed.

Clutches

As part of the standard component library, MapleSim provides two clutch models: a standard, controllable friction clutch and a one-way clutch [**2**]. Frictional clutch models include a lookup table to define the friction coefficients as a function of the relative speed between the friction pad and the plate. Axial force on the plate(s) is provided by an input signal. Geometric considerations can also be entered, such as the effect of the inner and outer pad radii and number of pad/plate assemblies. In DCL, these models are expanded; clutch and brake models provide a real output port for the loss power and a Boolean output port to indicate clutch lock-up. There are also other formulation improvements that make DCL models perform better when used with fixed-step integrators usually encountered in real time applications and Hardware-in-the-Loop simulations. DCL also includes a dog clutch component.

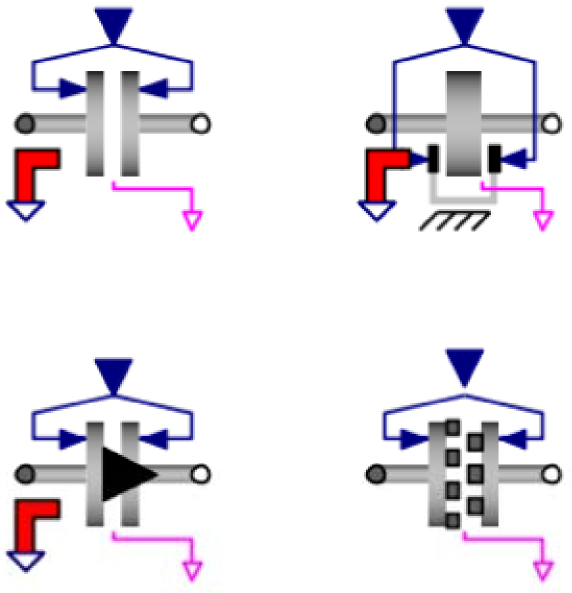


Figure 2. Brake and Clutch models in DCL

Torque Converter

The torque converter is modeled using tables of measured data. The following characteristics are used:

- Torque Ratio $\frac{\tau_t}{\tau_p}$ vs Speed Ratio $\frac{\omega_t}{\omega_p}$
- Load Capacity C vs Speed Ratio $\frac{\omega_t}{\omega_p}$

Where subscripts “t” and “p” designate turbine and pump quantities, respectively. The required data can be given as tabulated data or – as shown in Figure 3 – can be provided from external calculations via real signal ports. The Torque Converter component supports two alternative formulations based on the following definitions of the load capacity:

$$C = \frac{\tau_p}{(\omega_p)^2}\ \left[\frac{\text{N.m}}{\text{RPM}^2}\right] \text{ or } C = \frac{\omega_p}{\sqrt{\tau_p}}\ \left[\frac{\left(\text{rad}/\text{s}\right)^{1/2}}{\text{N.m}}\right]$$

Backward flow, happens during deceleration of the vehicle where the vehicle kinetic energy is transmitted back through the transmission to the engine. In this situation, the turbine is pumping and the pump is acting as a turbine. Since torque converters are not designed to work optimally this way, the torque converter will have very different characteristics. This is accommodated in the lookup table data by providing torque ratios and capacity values for $\frac{\omega_t}{\omega_p}$, typically up to about 5.

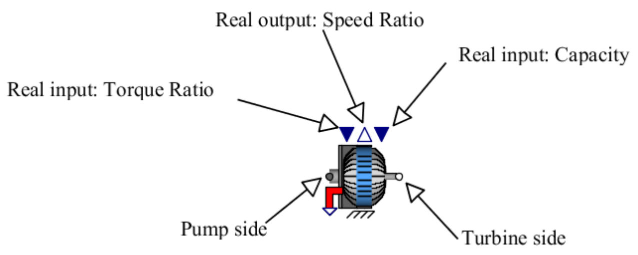
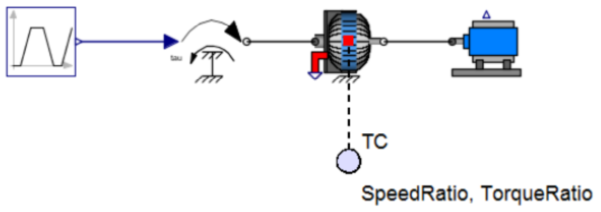


Figure 3. External data mode of the Torque Converter Component

In the test model shown in Figure 4, the input (pump) torque is increased linearly for the first 10 seconds. At low speeds, between *t* = 0 and 4 s, the turbine torque increases faster than the input torque. This is the "torque multiplication" effect typically seen in the torque converters [**3**]. Due to the inherent inefficiencies in the mechanism, the turbine speed is slightly less than the pump speed while the torque is driving the pump.



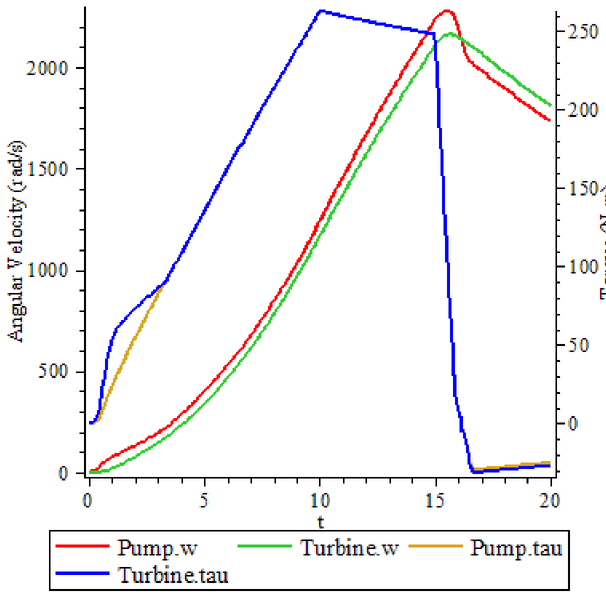
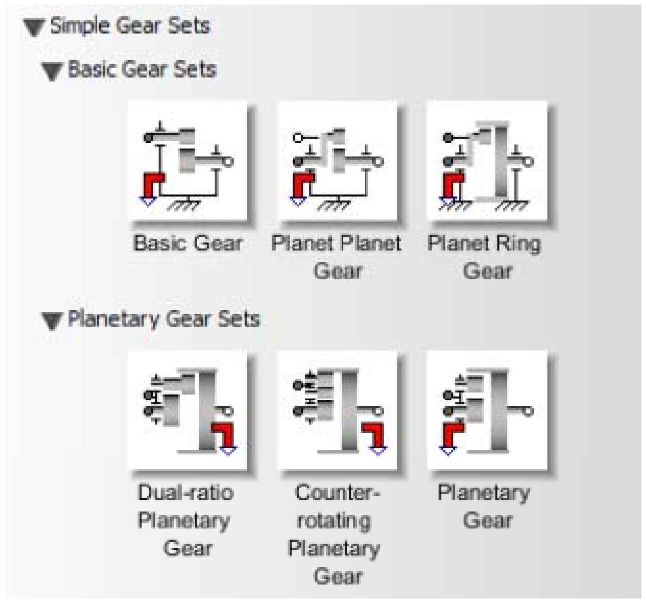


Figure 4. Torque converter test model

Note that when the input torque drops at t = 15 s, the kinetic energy of the dynamometer changes the torque flow from forward to backward (i.e. turbine drives the pump), and the pump speed drops below the turbine speed.

Gears, Gear Sets, and Transmissions

As shown in Figure 5, DCL includes simple and compound gear sets and related actuation components for modeling gear trains and transmissions. The Ravigneaux gear set component is discussed in the following as an example of the compound gear components in DCL.



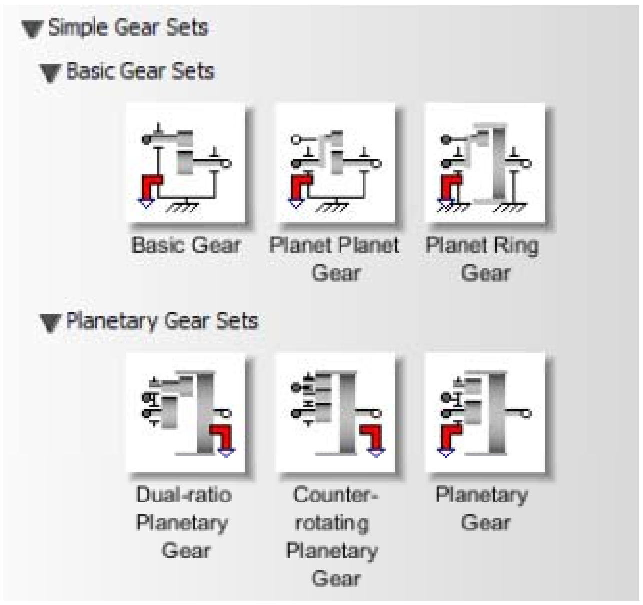


Figure 5. Collection of components for creating gear trains and transmissions

Ravigneaux Gear Set

The Ravigneaux configuration is a basic automatic transmission planetary assembly. As shown in Figure 6, this gear set includes two sun gears; a large sun gear and a small sun gear. These gears mesh with two planet gears connected to a rotating carrier. The meshing inner and outer planets rotate independently of the carrier. The inner planet meshes with the small sun gear and the outer planet meshes with the large sun gear. The ring gear then meshes with the outer planet and couples the gear-train together.

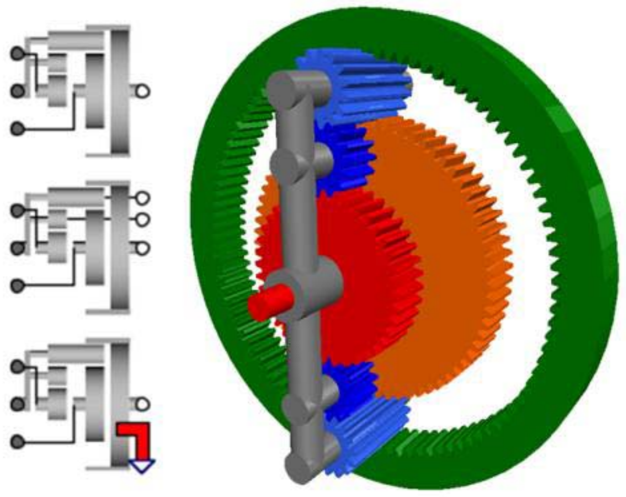


Figure 6. Ravigneaux Gear set

In DCL, this configuration is constructed internally using three Planet-Planet components and one Planet-Ring component, as shown in Figure 7. Ring/Small Sun and Ring/Large Sun gear ratios are provided by the user.

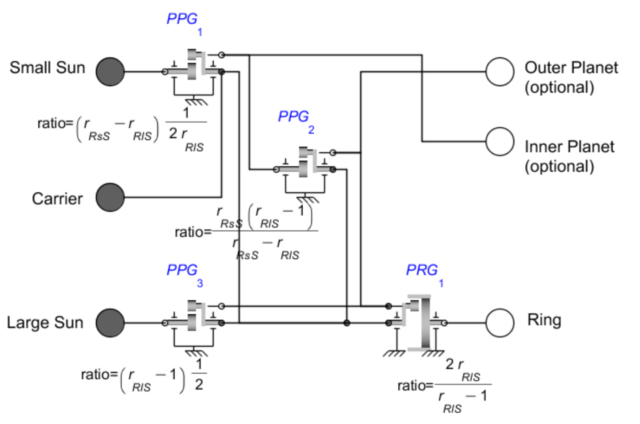

PPG_1
Small Sun
$\text{ratio}=\left(r_{RsS}-r_{RIS}\right)\frac{1}{2\,r_{RIS}}$
Carrier
PPG_2
$\text{ratio}=\frac{r_{RsS}\left(r_{RIS}-1\right)}{r_{RsS}-r_{RIS}}$
PPG_3
Large Sun
$\text{ratio}=\left(r_{RIS}-1\right)\frac{1}{2}$
PRG_1
$\text{ratio}=\frac{2\,r_{RIS}}{r_{RIS}-1}$
Outer Planet (optional)
Inner Planet (optional)
Ring

Figure 7. Internal structure of the Ravigneaux Gear

Selection of the required input/output transmission ratios is achieved by coupling or decoupling the mechanical ports by means of clutch components. The Ravigneaux Actuation component can be used as shown in Figure 8 to easily create a 4-speed transmission.

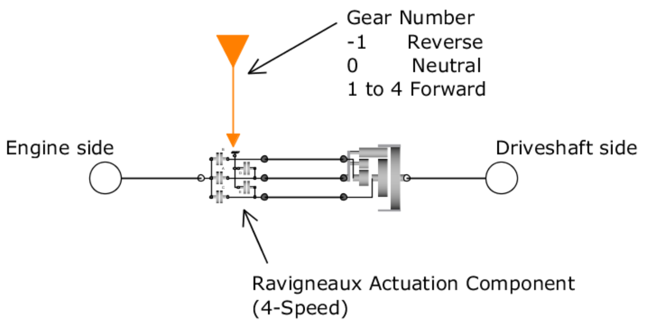
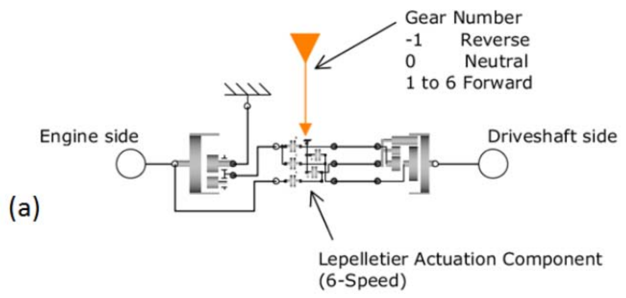


Figure 8. Building a 4-speed transmission with the Ravigneaux Actuation component

Lepelletier Gear Sets

There are two Lepelletier Actuation components (6-speed and 7-speed) provided in DCL which can be used together with a Ravigneaux gear and a planetary gear to create 6-speed or 7-speed transmissions as shown in Figure 9-a and 9-b, respectively.



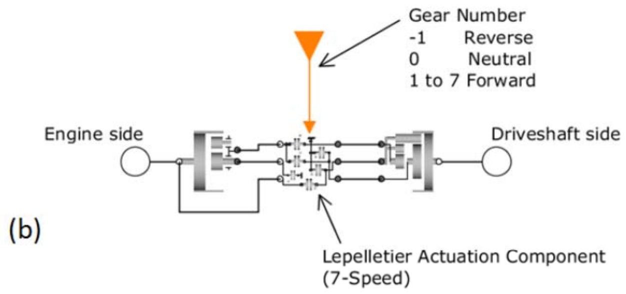


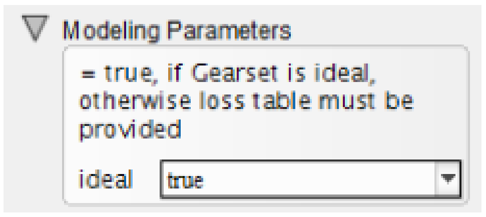
Figure 9. Building 6-speed (a) and 7-speed (b) transmissions with the Lepelletier Actuation components

Incorporating Losses

DCL contains all the mechanical building blocks necessary to build a driveline system model to the required level of detail. To achieve improved levels of fidelity, the library also includes lossy versions of all the basic components. The efficiency values can either be the results of calculations performed elsewhere in the model, or the loss factors interpolated from tables of measured data.

Meshing Efficiency

As shown in Figure 10, all of the gear components in the DCL can easily be switched from ideal (i.e. no losses) to lossy where power losses due to tooth-meshing are accounted for [**4**]. Incorporating empirical loss data allows users to make design decisions to reduce power losses and improve fuel efficiency based on the in-depth system-level insight.

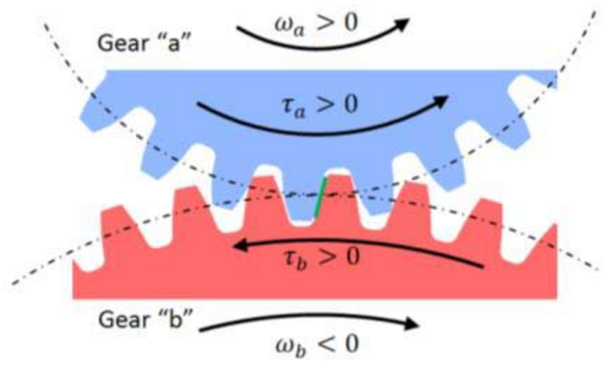


= true, if Gearset is ideal, otherwise loss table must be provided

ideal true

Figure 10. Fundamental GUI option for all gears – ideal = true/false

In lossy mode, the meshing friction is expressed as a transmission efficiency $(\eta(\omega) = \tau_{out}/\tau_{in})$ which may be defined as a function of the gear angular velocity via data tables. As shown in Figure 11, forward (leading teeth edge contact) and backward (trailing teeth edge contact) efficiencies may be defined independently [**4**].



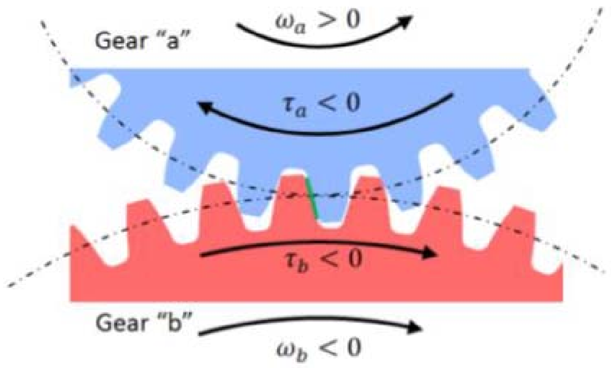


Figure 11. Gear Mesh – Leading/trailing edge contact

Bearing Friction

In compound gear sets (Planetary Gear, Dual-ratio Planetary Gear, Counter-rotating Planetary Gear, Ravigneaux Gear, Simpson Gear, and CR-CR Gear), internal bearing damping can be added using the component options. Bearing friction can also be added using external Bearing Friction components. Figure 12 shows how this is done for a Counter-rotating Planetary Gear component. The bearing friction is expressed as a torque loss and is related only to the shaft speed [**2**].

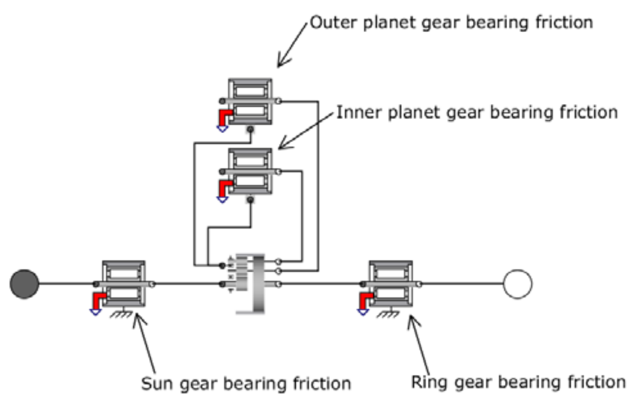

Outer planet gear bearing friction
Inner planet gear bearing friction
Sun gear bearing friction
Ring gear bearing friction

Figure 12. Adding bearing friction to gear sets

Losses in the Continuously Variable Transmission

A continuously variable transmission (CVT) allows a continuous range of gear ratios without the discrete gear changes necessary in other configurations. In DCL, this component can be switched to non-ideal mode where belt slip and torque loss is considered via experimental data tables. Assuming side "a" is the driver (i.e. $\omega_a \cdot \tau_a \geq 0$), the input/output torque relationship in the non-ideal mode is given by,

$$\tau_b = -\frac{1}{r} . \eta . \tau_a$$

where is the CVT base (geometric) ratio and is the efficiency (provided via a real signal port). Also the input/output speed relationship is given by,

$$\omega_b = r . \kappa . \omega_a$$

where $\kappa = 1 - s_r$ is the slip ratio and it is interpreted as,

$$s_r = \frac{r . \omega_a - \omega_b}{r . \omega_a} > 0$$

where s_r is the slip number and provided via a real signal port.

Figure 13(a) shows a simple model that incorporates a non-ideal CVT. The CVT ratio (reduction) is changed continuously from 1 to 2. The input and output power to and from CVT is plotted in Figure 13 (b). The effect of slip is shown in Figure 13 (c).

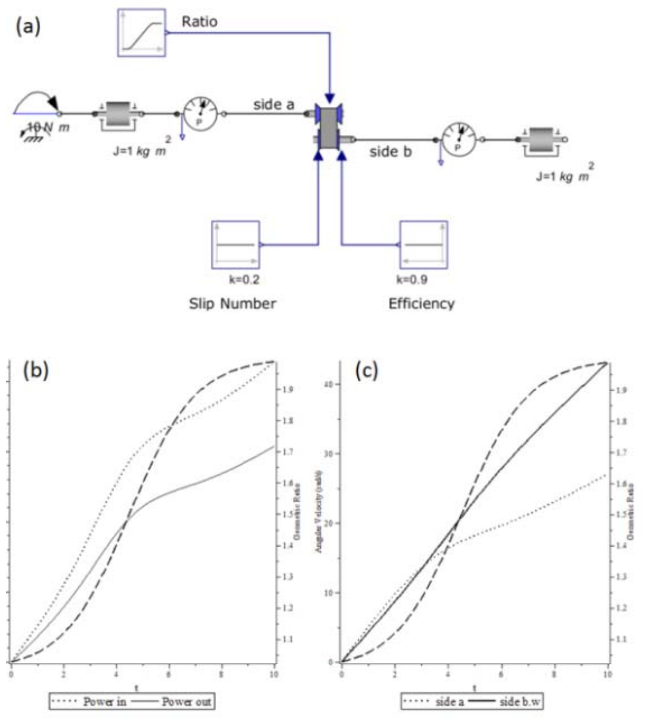


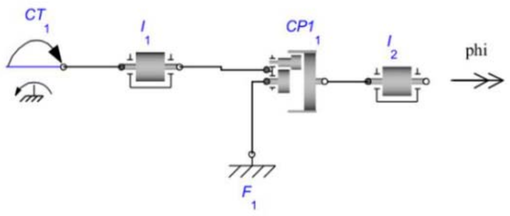
Figure 13. Non-ideal CVT component in DCL3

ADVANTAGE OF THE SYMBOLIC TECHNOLOGY

Symbolic techniques turn out to be a critical ingredient, both to enable efficient modeling of these components as well as to generate optimized code, yielding the required HiL execution speed. MapleSim's symbolic capabilities are enabled by an underlying Maple computation engine [6], providing extremely efficient symbolic operations that are necessary for handling the thousands of system equations typically found in the transmission models described in this paper.

An additional benefit to the symbolic approach is that since the entire set of equations for the system is explicitly generated, these equations can be manually inspected and used for mathematical analysis, in addition to generating simulation results and HiL code. The Maple environment is particularly well suited for such analysis at the equation level, given a system model. Figure 14, shows a simple example where the equations of motion are automatically generated in Maple.

(a)



(b)

$$\left[\begin{array}{c} \frac{d}{dt}\,\omega(t) = \frac{T\,(r2\,r1 + 1)\,r2\,r1}{r2^2\,r1^2\,J1 + J2\,r2^2\,r1^2 + 2\,J2\,r2\,r1 + J2} \\ \frac{d}{dt}\,\phi(t) = \omega(t) \end{array}\right]$$

Figure 14. (a) A simple model using Dual-ratio planetary gear, (b) Automatically generated equations of motion

MapleSim is a Modelica environment and the Driveline Component Library is developed using the Modelica language [**2**]. A common characteristic of Modelica environments is that system models are built by assembling components using “physical” connections, carrying quantities like torque and rotational angle bi-directionally between the two components. The decision on causality of the model is deferred to simulation time, just before the numeric integration process is started. This is possible because the entire set of equations for the whole system is generated symbolically, as a first step. At this point we typically have a set of differential algebraic equations. As shown in Figure 15, several steps are necessary before these equations can be solved numerically, yielding simulation results and/or HIL code. These steps are discussed next.

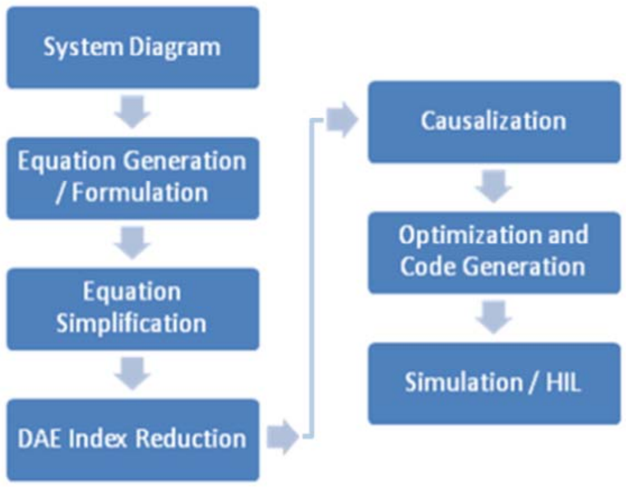


Figure 15. (a) A simple model using Dual-ratio planetary gear, (b) Automatically generated equations of motion

Equation Simplification

The initial set of equations generated from the system model is typically large and contains many redundancies. Symbolic techniques are used to simplify this set of equations as much as possible. The simplifications are exact and do not result in any loss of fidelity in the model. Trivial equations of the form a = b are removed. Linear equations are pre-solved analytically. Reducing the number of equations by a factor of ten is not uncommon. This simplification step is key to the scalability of the remaining pre-processing steps.

Index Reduction

As mentioned above, the generated system consists of differential algebraic equations (DAEs). Such equations cannot be readily solved with standard numerical techniques because of the presence of algebraic constraints. The “index” of a DAE is loosely defined as the number of times the equations need to be differentiated in order to remove these constraints. The goal here is to reduce the system of equations to “index 1”, allowing numeric integration. During integration, the constraints are monitored for “drift”, ensuring an accurate solution, reflecting the behaviour of both the differential equations as well as algebraic constraints. Again, symbolic techniques turn out to be essential, allowing differentiation of equations and efficient index reduction.

Causalization

At this point, we have a simplified system of (index 1) differential equations. In order to numerically solve this system, we will need to repeatedly evaluate the system for a particular point. To enable this, we will need to turn our (acausal) system of equations into a (causal) sequence of numeric operations. In short, this process involves imposing an order of evaluation onto our set of equations. Doing this efficiently involves tools from graph theory, readily available in the symbolic computing tool chest.

Optimized Code Generation

Executing speed is critical to HiL applications and symbolic techniques again turn out to be key to generating highly efficient code. It is, of course, possible to generate code directly from the causal system of equations described above. However by looking at those equations globally, we are able to perform symbolic optimizations prior to generating code, which makes the difference between achieving the required HiL cycle times or not. These optimizations involve detecting common computation sequences that can be factored out, which go way beyond the (local) optimization capabilities of available compilers.

Two Examples from DCL Models

<u>A Simple Driveline Model</u>

Consider the driveline mode shown in Figure 16. The model represents a vehicle powertrain from engine to dynamometer. The model includes a torque converter between the flywheel and the transmission. The transmission is a 4-speed Ravigneaux gearbox. Using throttle and brake controllers, the speed is changed following a ramp-up/coast down profile.

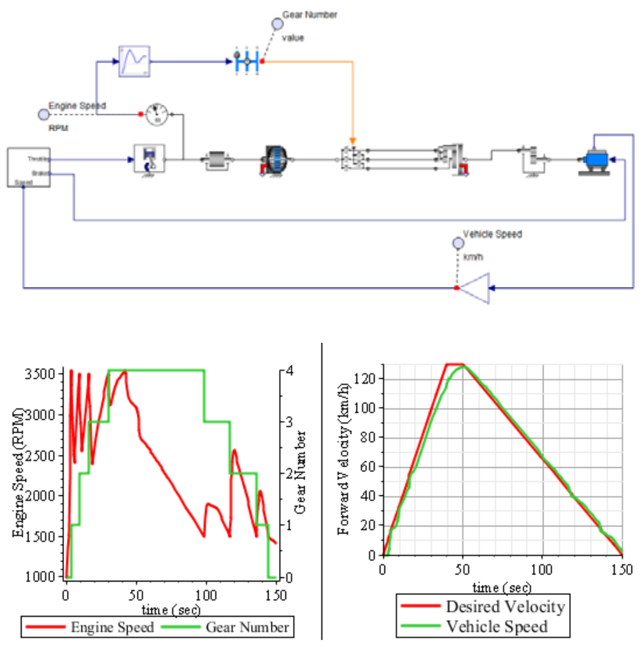
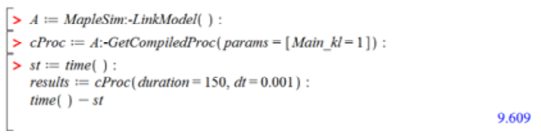


Figure 16. An example of a complete powertrain. The Ravigneaux gearbox is activated by a Ravigneaux Actuation component. The angular velocity of the load (Dynamometer) is controlled by two PID controllers through the throttle and brake commends.

Using MapleSim's API commands from Maple, the simulation time is measured. A fixed time-step solver (Euler) is used here with a time step of 0.001 sec. Total simulation time is 150 seconds. The simulation was done on a 64-bit Windows 7 machine with Intel(R) Core(TM) Duo 2.40 GHz CPU. Figure 17 shows Maple's commands for this example. These command extract and simplify the model equations and convert them to optimized c-code. The simulation results are obtained from a Maple procedure which includes the complied c-code.



```
> A := MapleSim:-LinkModel( ) :
> cProc := A:-GetCompiledProc( params = [Main_kl = 1]) :
> st := time( ) :
  results := cProc( duration = 150, dt = 0.001 ) :
  time( ) − st
                                                    9.609
```

Figure 17. Running MapleSim simulation using API commands

The simulation was done over 15 times faster than real-time (i.e. ~10 second of integration time for a 150-second simulation). In 20 consecutive runs the average simulation time was 9.68 with standard deviation of 0.30.

Full vehicle Model with Mean-value Engine Model

The system in Figure 1, is the second example chosen for the real-time demonstration. This model is considerably more complex than the previous example and includes a detailed mean-value engine model [**7**] and a 4-speed transmission model. The MapleSim model uses the New York City Cycle [**8**] and runs for 600 seconds. Simulation timing was done under similar solver settings as the previous example. The same computer was also used. On average the simulation was done about 12 times faster than real-time (i.e. ~50 second of integration time for a 600-second simulation). Based on 15 consecutive runs the average simulation time was 50.2 seconds with standard deviation of 0.54.

HIL SIMULATION OF THE AUTOMATIC TRANSMISSIONS

In this section some of the results obtained at AISIN AW Co., LTD in modeling automatic transmissions and HiL simulations are briefly discussed. At AISIN AW, HiL simulation is extensively used to accelerate the development of automatic transmissions. The plant models for HiL simulations require sufficiently high fidelity to accurately represent the aspect of the system dynamics important to the designers. At the same time, these models have to have low calculation cost in order to enable real-time execution. After a formal evaluation of available software, MapleSim modeling and simulation software together with the Driveline Component Library was chosen by AISIN AW to create real-time capable gear train plant models for HiL simulations.

Configuration of HiL system

As shown in Figure 18, the real-time platform used in the HiL simulations reported here is the ADX system [9] from A&D Technology, Inc. The plant model is deployed in Simulink [10] and can be separated into two parts as depicted in Figure 19. The first part is the plant model which is constructed of the s-function generated from MapleSim models including clutches, brakes, and various gear sets. This part also includes Simulink blocks for other parts of plant model. The second part is the automatic testing module.

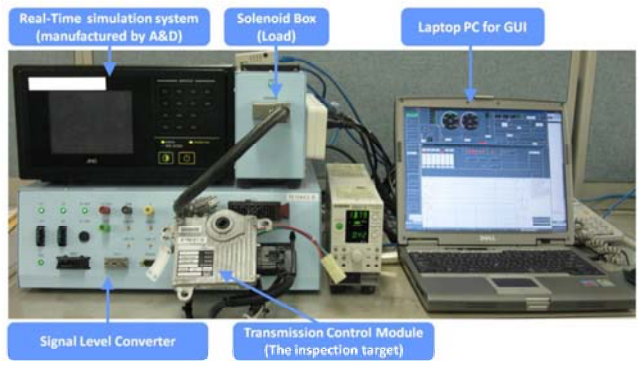

Real-Time simulation system (manufactured by A&D)
Solenoid Box (Load)
Laptop PC for GUI
Signal Level Converter
Transmission Control Module (The inspection target)

Figure 18. HiL simulation system

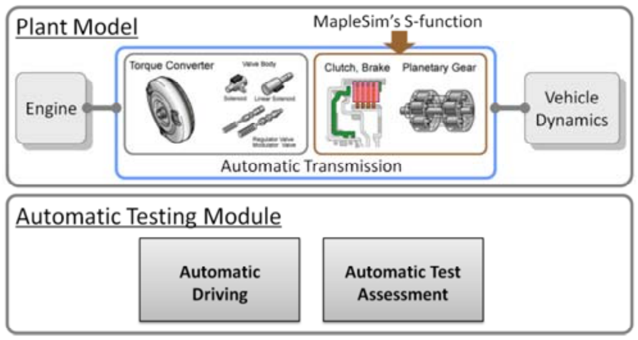


Figure 19. Model for Real-time system

It is critical that the calculation time associated with the first part (plant model) is kept as low as possible to accommodate for the high execution times of the increasingly more complex automatic testing routines implemented in the testing module.

One of the automatic transmission models - created in MapleSim - is shown in Figure 20. The gear train in this model includes a planetary gear, a Ravigneaux gear, and a basic gear connected together using three clutches, two brakes (modeled using clutch components), and a one-way clutch. This gear train is connected to an idear gear which represents the differential gear ratio. The tire load is modeled using additional inertia, clutch, and brake components. The tire component and the longitudinal vehicle dynamics component of MapleSim Driveline Component Library (refer to Figure 1) are not used here since that level of fidelity is not necessary for the intended HiL simulations. The s-function generated from the MapleSim's gear train model is integrated with other parts in Simulink.

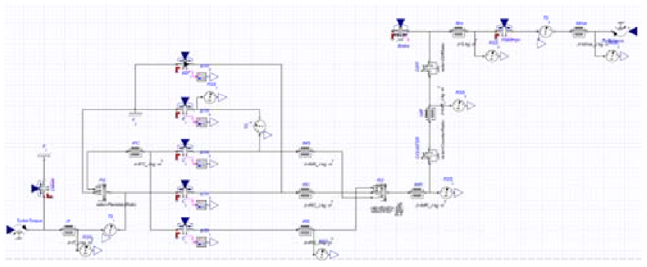


Figure 20. Gear train model created in MapleSim

The Effectiveness of Maple/MapleSim

Previously, AISIN AW used s-function generated from another simulation tool for the gear train model. Increasing demands to perform more complex testing routines necessitated a move towards modeling and simulation environments that could break the barriers of simulation execution time. In the following, some of the simulation results comparing MapleSim with the other simulation tool are presented.

The first step was to compare simulation results between the two software in off-line and real-time simulations. Figures 21 and 22 compare the results obtained from off-line gear train simulations in MapleSim and the other simulation tool. In these simulations The EPA Urban Dynamometer Driving Schedule (UDDS) [**8**] is used. The inputs to clutches and brakes as well as external torques were taken from previous HiL simulation data. Identical results obtained from the two software for the transmission input speed (Figure 21) and for the output speed (Figure 22).

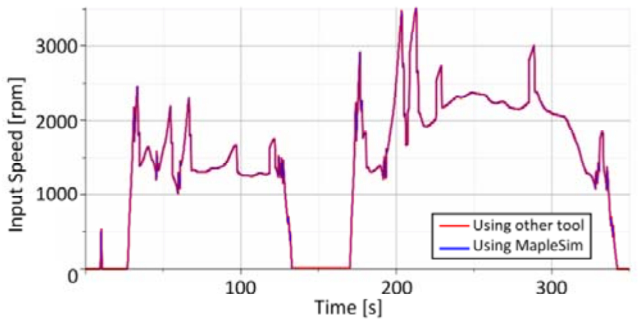


Figure 21. The results comparison of input speeds

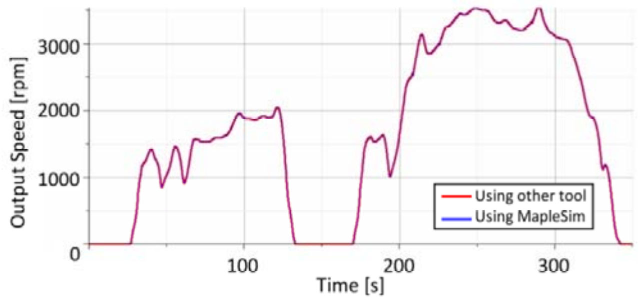


Figure 22. The results comparison of output speeds

After successful off-line tests, the numerical results from HiL simulations were also compared. Figures 23(a) and (b) show the HiL simulation results with s-function generated from the other tool and MapleSim, respectively. These simulations and many others demonstrated that both software produced the same results.

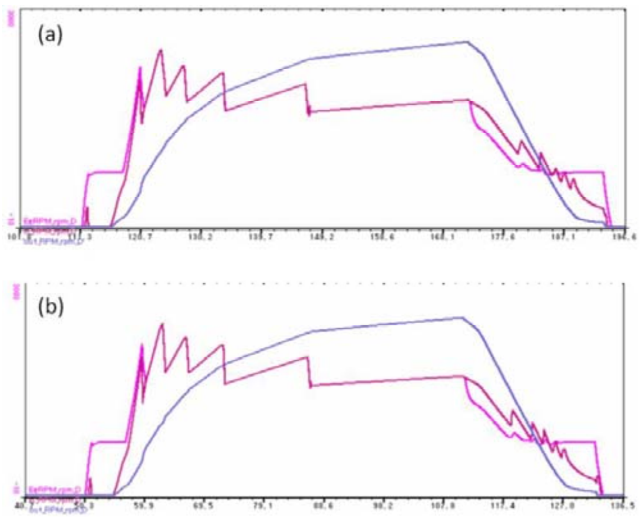


Figure 23. (a) HIL simulation result using the other tool's s-function (b) HIL simulation result using MapleSim's s-function

The next step was to compare the calculation costs in HiL simulations. Comparisons where done on identical platforms with a sampling time of 1ms. For the model shown above, the implementation of the s-function generated from MapleSim in the HiL simulations, reduced the overall CPU time by 250μ s (or 25%). This reduction translates to about half the calculation time required by the s-function generated by the other tool tested.

CONCLUSIONS

In this paper some of the features of the Driveline Component Library – an add-on library for MapleSim modeling, simulation, and analysis environment – were introduced. The Driveline Component Library provides a comprehensive set of components that enable transmission manufacturers – as well as other automotive developers – to conveniently create plant models for control and simulation. The underlying symbolic computation engine of MapleSim (i.e. Maple) expands the inherent advantages of similar Modelica-based physical modeling tools to new heights. Benefiting from the power of symbolic computing, MapleSim can generate extremely fast code that is of vital importance when simulating large complex systems in real-time. The paper also included a brief description of the activities at AISIN AW on the development of new automatic transmissions and their use of MapleSim and the Driveline Component Library in HiL simulations. The optimized c-code generated by MapleSim from transmission plant models enabled AISIN AW to perform more detailed HiL simulations. In a sample case study, it was shown that the s-function generated by MapleSim ran twice as fast as the s-function generated by a similar tool.

REFERENCES

[1] MapleSim User's Guide, 2011, ISBN 978-1-926902-09-8.
[2] https://www.modelica.org/ (accessed 2/4/2012).
[3] D. Hrovat and W.E. Tobler. "Bond graph modeling and computer simulation of automotive torque converters," Journal of the Franklin Institute. Volume 319, Issues 1-2, January-February 1985, pp 93-114.
[4] Pelchen C., Schweiger C., and Otter M., "Modeling and Simulating the Efficiency of Gearboxes and of Planetary Gearboxes," 2nd International Modelica Conference, Proceedings, pp. 257-266.
[5] Joško Deur, Vladimir Ivanovic´, Matthew Hancock, and Francis Assadian. "Modeling and Analysis of Active Differential Dynamics," Journal of Dynamic Systems, Measurement, and Control, 2010. Volume 132 / 061501, pp 1-14.
[6] Bernadin L., Chin P., DeMarco P., Geddes K. O., Hare D. E. G., Heal K. M., Labahn G., May J. P., McCarron, Monagan M. B., Ohashi D., and Vorkoetter S. M., Maple Programming Guide, 2011, ISBN 1-926902-08-1.
[7] - , "Mean-Value Internal Combustion Engine Model", Maplesoft, White Paper, http://www.maplesoft.com/contact/webforms/whitepapers/enginemodel.aspx, (accessed: 2/4/2012).
[8] -, Dynamometer Driver's Aid, http://www.epa.gov/nvfel/testing/dynamometer.htm, (accessed: 2/4/2012).
[9] http://www.aanddtech.com/ADX.html (accessed: 2/4/2012).
[10] http://www.mathworks.com/products/simulink/ (accessed: 2/4/2012).