Practical Formal Verification for Model Based Development of Cyber-Physical Systems

Tasuku Ishigooka
Center for Technology Innovation – Controls,
Research & Development Group, Hitachi Ltd.,
Ibaraki, Japan
tasuku.ishigoka.kc@hitachi.com

Habib Saissi, Thorsten Piper, Stefan Winter,
Neeraj Suri
Technical University of Darmstadt,
Darmstadt, Germany
{saissi, piper, sw, suri}@cs.tu-darmstadt.de

***Abstract*— The application of cyber-physical systems (CPSs) in safety-critical applications requires rigorous verification of their functional correctness and safety-relevant properties. We propose a practical verification framework which enables to fill the gaps between model-based development and the formal verification process seamlessly connecting them. The verification framework consists of (a) a model transformation method, which automatically transforms the plant models of CPSs including differential algebraic equations (DAE) to equivalent models without DAE to reduce verification complexity induced by DAE solver execution, and (b) a model simplification method, which automatically simplifies bond-graph models by replacing complex bond-graph components with simpler components for further verification overhead reductions.**

We successfully applied the proposed verification framework for safety verification of an automotive brake control system. The results of the study demonstrate that the automated model transformations of the CPS models yield significant verification complexity reductions without impairing the ability to detect unsafe behavior of the brake control system in a formal verification based on symbolic execution.

Keywords*— *Formal verification, model-based development, model transformation, bond-graph model, signal-flow model, cyber-physical systems

I. INTRODUCTION

A cyber-physical system (CPS) is an embedded control system that strongly links computing and physical systems [1][2]. For example, automotive safety-critical CPSs consist of controllers (cyber), called electronic control units (ECU), and control targets, such as mechanical components (physical), called plants. The ECU software measures plant behavior through sensors and controls actuators by issuing control commands in real time in accordance with the sensed state of the plant. Automotive safety-critical CPSs implement a highly collaborative control process between electronic and mechanical components.

Automotive CPSs have stringent safety requirements, because system failures may cause critical damage to users. Therefore, the development process for CPSs requires rigorous verification steps. In automotive CPSs, the model based development (MBD) approach prevails. The approach requires controller models, which execute discrete processing, and plant models, which have continuous behavior based on the physical laws, as shown in Figure 1. Concretely, the controller model comprises control algorithms described by ordinary differential equations and the plant model replicates physical control target behavior described by differential algebraic equations (DAE) for the energy conservation theorem, in which power of action to the physical object and power of reaction from the object are equal.

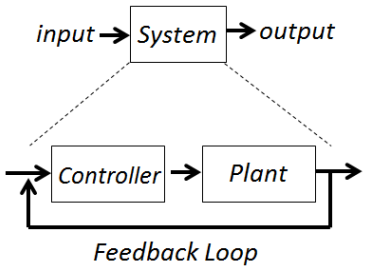

Figure 1. Control system

There are modeling tools that aid the design of controller or plant models. For example, the controller model design is commonly conducted by signal-flow diagram based modeling tools such as MATLAB/Simulink®[3]. For the plant model design, bond-graph modeling tools such as AMESim™[4], Simscape™[5], and Modelica® [6] -compliant tools are predominant.

These models are used to validate the system design of automotive CPSs prior to building an actual prototype. The system behavior is simulated by numeric solvers, such as ordinary differential equation (ODE) solvers and DAE solvers. However, it is difficult for users to identify subtle design faults which occur only upon rare combinations of specific conditions, because these constitute a vanishingly small fraction of all possible test cases. This makes the selection of such cases in the testing process very unlikely. Nevertheless, if these rare conditions occur during operation of the system, any unidentified defects can severely threaten the safety of its users.

There exist various formal verification approaches based on hybrid system checking [7][8][9][10][11]. Unfortunately, these approaches have not seen wide spread adoption in industry, as the creation and verification of suitable hybrid system models in real mass production projects requires extensive knowledge of formal verification by engineers, who

mostly have an exclusive mechanical engineering background. In order to address that, there are practical formal verification approaches which comply with model-based development processes [12][13][14]. These approaches reduce hybrid system modeling efforts for the engineers. However, these approaches only work with systems based on ODE.

Unfortunately, for a large class of safety-critical CPSs solely ODE-based models are not sufficiently expressive. In many cases models of these systems contain DAE to reflect the energy conservation theorem. In numerical simulation, the plant behavior is simulated by leveraging a DAE solver. The DAE solver is executed at every calculation step in the simulation in order to find a set of suitable values of specific variables, such as a set of action and reaction forces by convergence calculation. Thereby, the DAE solver enables correct physical simulation but produces excessive computation load, which complicates automated verification beyond practicability.

While approaches exist to remove DAE by formula manipulation of partial differentiation and substitution, they are difficult for engineers to manually apply without introducing errors into the transformed models and, thereby, threatening the validity of the verification. Thus, we propose a practical verification framework for safety-critical CPSs, which are modeled by bond-graphs, and a model transformation method. In this approach, we transform bond-graph models with DAE into signal-flow models without DAE. To avoid the overhead for manually creating additional models for correctness and safety verification (as necessitated by hybrid automata based approaches), our proposed framework is capable of reusing plant models developed in MBD processes. Since the plant models are originally developed for sophisticated simulation purposes, the plant models are accurate and complex. As a consequence, the reuse of these complex plant models for verification purposes yields prohibitive computational complexity. In order to address the issue, we also propose a model simplification method complying with MBD. In summary, this paper makes the following contributions to the state of the art in CPS verification.

- A practical verification framework complying with MBD for cyber-physical systems
- An automatic model transformation method from bond-graph plant models with DAE into signal-flow models without DAE
- An automatic model simplification method to reduce computation load by domain-knowledge-based replacement of complex model components and approximation of the model behavior by model parameter configuration based on feedback of simulation results.

II. BACKGROUND

We start with an overview of mechanical electronic control systems in Section II-A, before we present the usage of bond-graph modeling in Section II-B.

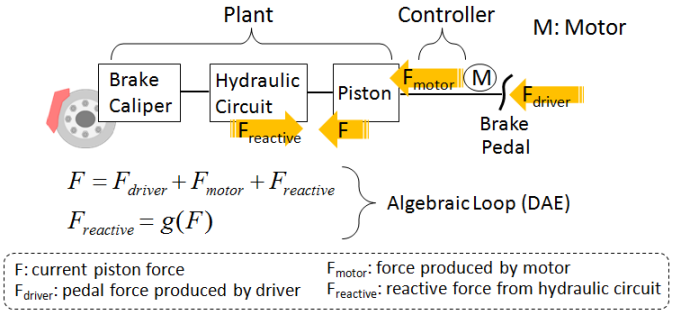

Figure 2. An example automotive brake system

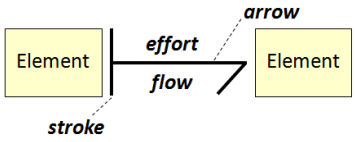

Figure 3. Notation of bond-graph

A. A Cyber Physical System Example

To discuss the specifics of CPSs modeling, we consider the simplified automotive brake control system presented in Figure 2. The brake control system produces a brake force in accordance with the moving amount of the brake pedal stroke operated by a driver. As shown in Figure 2, the system consists of mechanical components, such as brake calipers, which is a speed reducer, hydraulic circuits and motors, and electronic components such as ECUs controlling the motors. In the system, an ECU controls the brake force by monitoring the movement of the brake pedal stroke operated by the driver and assisting the pedal force by controlling the motor in synchronization with the moving amount of the stroke. The assistance enable users to lightly push the brake pedal.

The example of Figure 2 shows how current piston force F is determined. Current piston force F is the sum of the pedal force produced by the driver F_{driver}, force produced by the motor F_{motor}, and the reactive force from hydraulic circuit $F_{reactive}$. $F_{reactive}$ is the result of applying function g with argument F. These formulae reflect the law of energy conservation resulting in the plant of the brake control system being modeled as equations with an algebraic loop. As a consequence, the plant model includes DAE, which impose a challenge to automated or computer-assisted verification.

B. Bond-Graph Modeling

Bond-graph modeling enables users to design plant models by combining physical components such as spring, mass, and hydraulic circuit. Each component can be mapped to a real physical component and has a specific equation, such as a motion or constraint equation, and specific configurable parameters, such as weight, length, and so on.

As shown in Figure 3, a bond-graph model consists of elements which translate to components, effort, flow and stroke. Effort and flow are domain-independent in bond-graph notation. For example, in the mechanical domain, effort means force and flow means velocity. In the hydraulic domain, effort means pressure and flow means volume or flow rate. Stroke means flow direction. In the example, the flow value

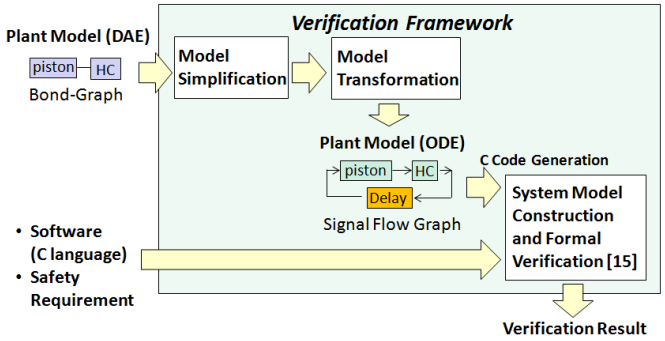

Figure 4. An overview of our proposed verification framework for CPS

calculated by the left element is an input to the right one. The stroke reflect the causality in the calculation order of equations of each component. Finally, arrows describe energy direction.

III. Our Proposed Approach

We present a verification framework, automatic model transformation method and automatic model simplification method in Section III-A, Section III-B, and Section III-C, respectively.

A. Verification Framework

We propose a verification framework which enables the safety verification of CPSs. Figure 4 shows an overview of the framework, which consists of a model simplification phase, a model transformation phase, and a phase for system model construction and formal verification.

As Figure 4 shows, a bond-graph plant model with DAE, which is an input file, is automatically simplified at the model simplification phase, which is presented in Section III-C, and then is transformed into the signal flow plant model with ODE at the model transformation phase, which is presented in Section III-B. After our framework obtains the transformed model with ODE, it leverages the framework proposed in [15] to efficiently enable to verify the control system. The framework from [15] comprises a system model construction phase, where the virtual system model is constructed by combination of software, safety requirement, and the plant code which is converted from the plant model with ODE, and a formal verification phase, where automated formal verification, such as symbolic execution, is applied to verify the safety of the system model.

The purpose of safety verification is to prove CPS safety in specific situations where potential safety violations might occur. The specific situations means system state transitions, such as a state transition, in which a driver strongly pushes a brake pedal immediately after the brake pedal was released. We assume that the verification engineers verify individual safety properties by using the verification framework from [15]. Although the verification framework employs bounded state search algorithm, the verification is conducted in carefully chosen internal simulation time, which can cover the duration of the specific situation.

In our verification framework, we propose to do the model simplification before the model transformation because it is

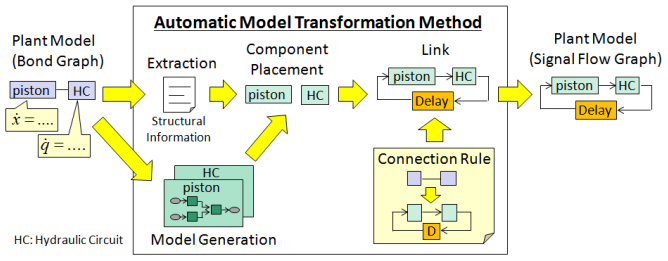

Figure 5. An overview of automatic model transformation method

easier to simplify the bond-graph plant model than the signal-flow plant model.

B. Model Transformation Method

Elimination of DAE from bond-graph models is known to be difficult because it entails the elimination of energy exchange in the bond-graph models which follows the law of conservation of energy. There is an approach in which the engineers design ODE model by manually transforming the DAE models, which requires deep knowledge about formulae translations in physics. We assume that software engineers take over the verification process and conduct the safety verification as part of the system test process. Therefore, the engineers conducting safety verification cannot be expected to have the intimate knowledge about physics required for manual model transformation. Thus, we propose a model transformation method, which transforms the bond-graph model to the signal-flow graph model and converts the model with DAE to ODE by adding delay blocks in between blocks with energy exchange.

A simple model transformation would replace a bond graph model with an equivalent signal flow model. However, the transformed model will still contain the algebraic loop from bond-graph model. Our approach can eliminate the algebraic loop by inserting one-step-delay blocks into feedback loops between components. The purpose of delay blocks is to make the connected block use the signal value generated at the previous period. In the first calculation step, the delay block produces an initial value defined by the user. The initial value should be copied corresponding parameters of the components calculated at the beginning of the specific situation of simulation. Consequently, our proposed method can remove the algebraic loop from the plant model.

The delay block produces calculation errors at every simulation step. However, our evaluation results presented in Section V-A show that these errors remain negligibly small for the short simulation times commonly required for the assessment of individual safety properties.

In order to reduce the effort and avoid human errors during the model transformation, we propose an automatic model transformation method which automatically transforms bond-graph plant models to signal-flow plant models. Figure 5 shows an overview of our automatic model transformation method. This method automatically analyzes input files which store bond-graph plant models and extracts structural information on the plant models. In parallel, the method generates signal-flow subsystems according to the equation of each element in the bond-graph model. Each subsystem

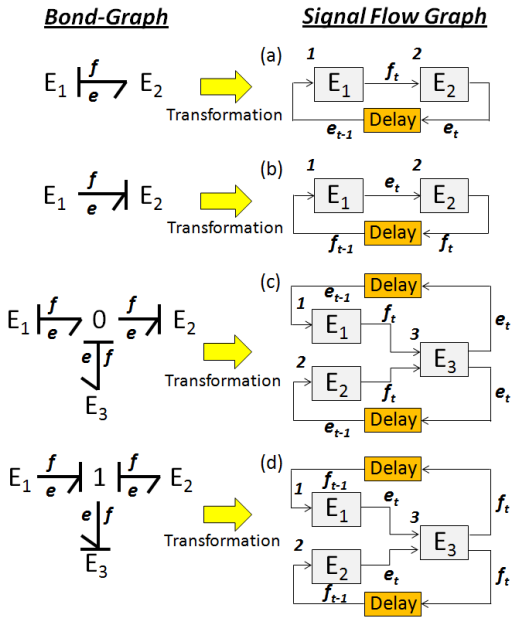

Figure 6. Connection rule

corresponds to exactly one element in the bond-graph model. The method places these subsystems according to the previously extracted structural information. Hence, it links each subsystem according to our proposed connection rule. Ultimately, this method outputs signal-flow graph plant models.

We defined connection rules of signal-flow-graph subsystems to preserve semantic equivalence with bond-graph component interactions. Figure 6 shows our proposed connection rules. In bond-graphs, there are two data types, flow and effort. Moreover, there are two connection types, direct and multiple. The direct connection connects one element to another. The multiple connection involves more than 3 elements. The connection is implemented by a 0 junction or a 1 junction (see Figure 6 (c) and (d)). The connection rule (a) is used for the direct connection situation when E_1 outputs flow signals to E_2 and E_2 feedbacks effort signals to E_1. The effort signals are delayed by one period through a one-step-delay block. The connection rule (b) is used in the opposite situation. The connection rule (c) is used for the multiple connection situation when E_1 and E_2 output flow signals to E_3 and E_3 feedbacks effort signals to E_1 and E_2. The connection rule (d) is used in the opposite situation from (c).

C. Model Simplification Method

We assume that the bond-graph plant models, which are originally developed in MBD for simulation purposes, are reused for verification purposes. These plant models are highly sophisticated because they are designed to check whether the controller model (see Figure 1) meets functional and real-time requirements such as the increase of a parameter value by a specific amount within specific time. Unfortunately, formal verification approaches are commonly very sensitive to the complexity of the verification target. Luckily, the verification of safety properties can commonly be conducted

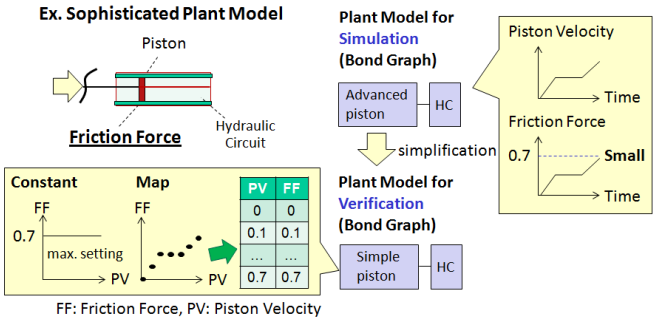

Figure 7. An overview of simplification approach using an example of sophisticated plant model

on dramatically less complex models that overapproximate the original model's properties. However, as we mentioned in Section III-B, most verification engineers do not have the deep knowledge of physics or plant modeling in MBD that is needed to create such sound overapproximations for verification. As a consequence, the construction of additional simpler models for the sole purpose of verification does not only require redundant work, but it is also error-prone if conducted by engineers who have limited experience in crafting such models. Therefore, we propose an automated simplification method for bond-graph plant models to reduce computation load without requiring manual model redesign by verification engineers.

The idea of the simplification approach is to find elements, which produce excessive computational load but have little impact on verification-relevant parameters and replace them with functionally equivalent elements that create lesser computational load during verification. Figure 7 shows an overview of this simplification approach using an example of a sophisticated plant model. In this example, the plant model for simulation includes an advanced piston, which dynamically calculates friction force according to the current velocity. This dynamic calculation approach can provide highly accurate physical simulation.

Our proposed approach replaces advanced elements, such as the advanced piston, by simple elements, such as the simple piston. Which element should be replaced depends on the target system. For example, in a brake control system the advanced piston is a candidate element for replacement because the physical size of the piston and its amount of produced friction force are relatively small. In this case, a simple piston with a configurable constant friction force parameter is a viable substitute. The parameter is configured to approximate the plant behavior. The configuration is based on simulation results, which are calculated before the replacement procedure. Two examples for such configurations are (1) the constant configuration of a parameter value as its maximum assumed in simulation and (2) the definition of a mapping table for a series of values the parameter assumes in simulation depending on some other parameter. The two options are illustrated in the lower left of Figure 7.

Figure 8 illustrates the simplification approach by 1:1 element replacement. In this example, a sophisticated mass I_{so} of a mass spring model is replaced with a simple mass I_{si} The left bond graph model is equal to the right simplified mass

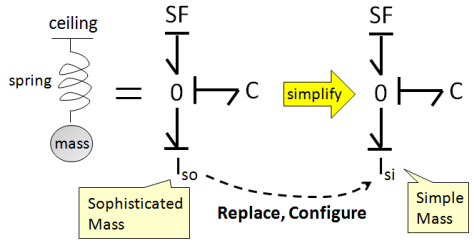

Figure 8. An example of simplification approach by 1:1 element replacement

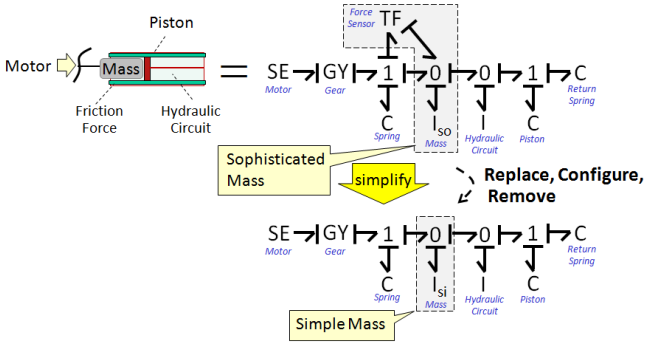

Figure 9. An example of simplification approach by N:1 element replacement

spring model at the symbolic level. Our proposed simplification method replaces an element by a simpler instance of the same element type such as I_{so} and I_{si} in Figure 8 and configures the parameter of the replaced element according to fixed parameters of the original elements (such as weight) and variable parameters, which are dynamically calculated. Dynamically changing parameter values are based on simulation results.

Figure 9 shows the simplification approach by N:1 element replacement. In this case we replaces target elements by the same procedure as 1:1 element replacement. Furthermore, we remove irrelevant elements. In this brake control system example in Figure 9, the simplification method replaces a sophisticated mass including a relative element with a simple mass. The velocity of sophisticated mass is measured by leveraging TF, which represents a force sensor to monitor mass force calculated on the basis of mass velocity, and dynamically calculates friction force of the mass according to its velocity. The friction force is used for calculation of precise mass force. For example, if we want to replace a sophisticated mass with a simple mass, the simplification method replaces I_{si} of the above bond-graph model. The method also removes TF as an irrelevant element because the simplified I_{si} uses fixed friction force instead.

The selection of suitable approximate elements for simplification requires deep application domain knowledge of the plant characteristics. Out automated method is based on domain expert knowledge so that verification engineers can utilize the element selection options without prior knowledge of physics.

Figure 10 shows our proposed automatic model simplification method. The replacement and configuration knowledge from experts is implemented as a replacement table. Our proposed method conducts automatic simplification

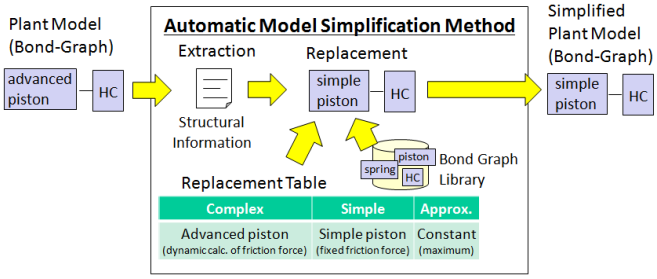

Figure 10. Automatic model simplification method

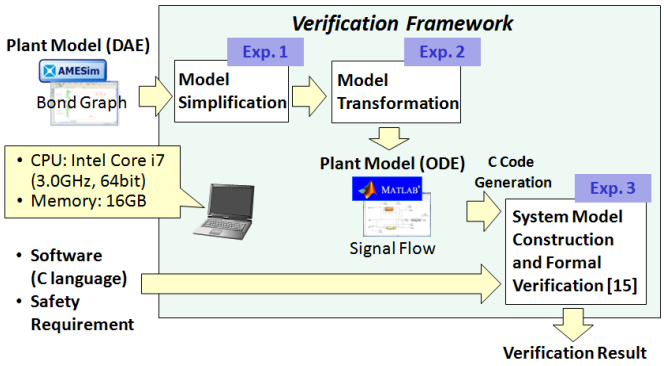

Figure 11. An overview of experiments

according to this table. The replacement table stores information on the relationship of complex elements to their simpler replacement candidates in the target system and information on the recommended configuration approach for variable parameters.

Our method extracts structural information from a bond-graph plant model stored in an input file. Then, the method conducts element replacement by identifying candidate elements for replacement in the bond-graph according to the replacement table and structural information, replacing them by simple ones, and setting the simpler elements' configuration parameters. In Figure 10, the maximum value of dynamic friction force observed in simulation results is used as a constant value of static friction force.

IV. Experiment

We conducted experiments to evaluate our proposed methods. In this section, we present our experiment environment, the obtained experimental results, and a discussion about how to interpret them.

A. Experiment Environment

Figure 11 shows an overview of the experiments. To validate the feasibility of the proposed methods, three experiments have been conducted. In the first experiment, we measured how much our model simplification method reduces computation load without tolerating any decrease of verification accuracy. In the second experiment, we measured to which degree our model transformation method introduces behavioral deviations between the bond-graph model and the signal-flow model. In the third experiment, we measured time for conducting automated safety verification of the transformed model and checked whether the verification

enable failure detection after our proposed transformation and simplification method. As a verification tool, we chose KLEE [16] with default configuration of the symbolic execution engine and the STP solver [16] for solving path constraints. In the first and second experiments, the operating system is Windows® 7 32-bit and in the third experiment it is Ubuntu® 12.04 32-bit. The first and second experiments ran on a machine with 2.80 GHz Intel® Core i5 quad-core processors and 2 GB of RAM and in the third experiment it with 3.0 GHz Intel® Core i7 quad-core processors and 16 GB of RAM.

We applied the above experiments to an automotive brake control CPS. We implemented our proposed model simplification method and model transformation method as two prototype tools. For the model transformation tool, the insertion of delay blocks has been conducted manually. Additionally, in the second experiment the model generation procedure, which is shown in Figure 5, is also conducted manually, in the sense that signal-flow subsystems for model transformation were developed manually before the model transformation tool execution. The tool makes use of these manually developed subsystems in the component placement procedure shown in Figure 5. As the manually identified (but automatically applied) subsystems are reusable for other models that contain the same subsystems, we consider this a one-time effort. For example, since the signal-flow piston subsystem is frequently used in many plant models of the same domain, we consider the subsystem replacements to be reusable *at least* for the same domain, i.e., the product family.

For these experiments, we developed a brake control system model, shown in Figure 2, which consists of a controller model implemented in C, and a plant model designed in AMESim. The plant model is discretized using 100 micro seconds intervals. This means that one-step delay blocks delay the target signals for 100 micro seconds. In the first experiment we measured the effect assessed by simulation and reused the plant model of the real mass production development with the same discrete time intervals instead of our developed model in order to measure the accurate effect.

We embedded a subtle fault, which results in unintended brake force, in the control software. In the third experiment, the verification must detect this fault by checking the resulting violation of the safety requirement, i.e., unintended braking does not occur. We implemented three conditions to detect the safety requirement violation as assertion code according to [15]. If all of these conditions are satisfied, the safety requirement is violated. The first condition is that the brake pedal is not actuated. The second condition is that the elapsed time is at least 500ms after the pedal released. The time prevents the verification tool from misdetections caused by the response delay of the plant. The third condition is that the amount of piston displacement, which means distance from initial piston position, increases, i.e., the brake force increases. A brake pedal operation, which is a system input, is defined as a symbol supplied by KLEE each second through symbol re-definition [15]. We generated the system model combined by the control software, the plant code, and the assertion code

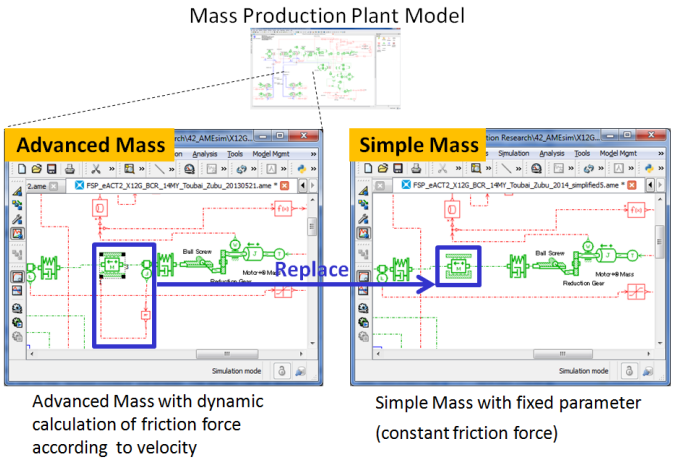

Figure 12. The first experimental result on the model simplification method

Table 1. Comparison results of simulation time

Plant Model	Average of Simulation Time [s]	Reduced Rate [%]
before simplification (advanced piston)	179	-
after simplification (simple piston)	116	35.2

[10]. The model replicates the system behavior during 5 seconds in internal time and is structured by 2000 lines in C. We verified the safety of the model according to [15].

B. *Experiment Results*

Figure 12, Table 1, and Figure 13 show the first experimental results of the model simplification method. The simplification tool found an advanced mass and replaced it to simple mass and configured it for approximation according the replacement table (see Figure 12). This single replacement yields approximately 35% computation load reduction in simulation (see Table 1).

Additionally, in order to compare the simplified plant model behavior with the original one, we plot their behaviors in Figure 13 for an easy visual comparison. As the result shows, there is no recognizable difference. This means our model simplification method did not adversely affect the accuracy of the model.

Figure 14 shows the second experimental result on the model transformation method. The model transformation tool could transform the bond-graph plant model of the brake control system described in AMESim into a signal-flow plant model for MATLAB/Simulink. The plotted simulation result shows the brake piston displacement of the original model (left) and the transformed model (right) with changing brake force. To achieve a sound comparison, we applied the same settings for the numeric solver method and configuration parameter in both AMESim and MATLAB/Simulink. As the result in Figure 13 illustrates, there was no recognizable error.

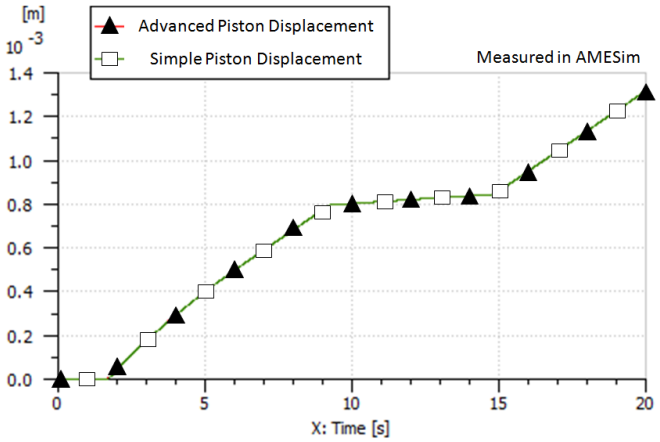

Figure 13. Comparison results of plant behavior before and after simplification

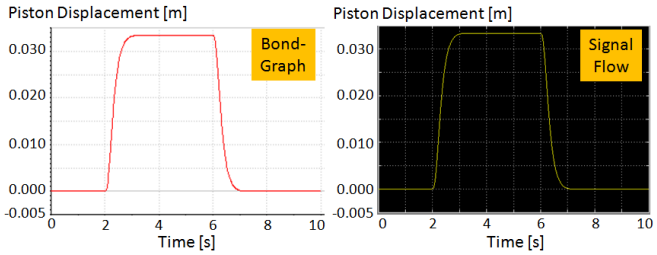

Figure 14. The second experimental result on the model transformation method

Figure 15 shows the third experimental result on the safety verification. In the upper part of the figure illustrating the driver brake pedal operation, ON indicates pedal actuation. OFF indicates pedal release. The piston displacement in the lower part of the figure shows, there is an increase (indicating brake engagement) despite pedal release between seconds 4 and 5. This means that unintended braking occurred even though the driver did not actuate the brake pedal. This failure was detected by spending 23 minutes 15 seconds of verification time. This failure condition is the expected result of the fault we embedded. Therefore, the result confirms that the verification tool was able to properly verify the safety of the system model including the plant model transformed by our prototype tool. After the failure was fixed, we conducted the safety verification. The safety was verified by spending 23 minutes 44 seconds of verification time.

V. Discussion

In order to facilitate the automated verification of safety properties in critical CPSs, we have proposed model transformations and simplifications. Obviously, none of the proposed modifications to the CPS models should affect the validity of the verification. In the following we discuss the effects of our modifications on the system behavior in simulation.

A. *Effects from the Model Transformation Approach*

Our model transformation approach inserts one-step delay blocks between components. In order to clarify the impact caused by the blocks, we measured each output signal value of the transformed plant model in Simulink in the case of one-

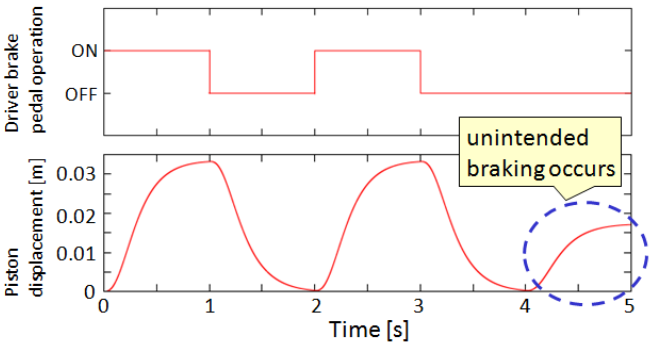

Figure 15. The third experimental result on the safety verification

step delay, which our approach inserts, and in the case of two-step delay for the comparison to extract the difference. We did not measure the signal value of the model with no delay blocks because the model did not work in Simulink due to the algebraic loops. As we mentioned earlier, one step means 100 micro seconds. The measurement was done at the same condition of the second experiment in Simulink. The result showed that the maximum difference of plant output signal was 0.1 %.

Bond-graph modeling enables us to design the plant model by connecting physical components. Our transformation method inserts a delay block into a feedback loop between two components. The block delays one simulation step such as 100 micro seconds to the value of the feedback. This means each component state is individually updated by the current input signal and the previous output signal as reaction value. Therefore, the delays caused by the blocks do not sum up. Consequently, the effect from our model transformation approach is acceptable.

B. *Effects from the Model Simplification Approach*

The purpose of our safety verification method is to check software logical error resulting in the safety violation from the view of the system behavior. In order to reduce the verification complexity of the plant model, our model simplification approach abstracts the target plant model. In general, the model abstraction approach may cause false positives. Our approach can reduce the complexity as preserving the number of the system state and the plant behavior. Therefore, our simplification approach provides a sound overapproximation of the system model.

VI. Related Work

There are related work for multi-domain model-based design and verification approach. The study enables users to ensure consistency between heterogenous models such as control models, plant models and process algebraic models [17] by developing a system architecture, in which these models are encapsulated, and each corresponding component is connected [18]. The architecture model maintains structural and semantical consistency on logical constraint of model parameters [19][20]. Moreover, the environment in [22] can generate verification models which are described by finite state processes (FSP) or liner hybrid automata (LHA) [21].

FSP models can be analyzed by Labelled Transition System Analyzer (LTSA) tool [17]. LHA models can be analyzed by Polyhedral Hybrid Automaton Verifier (PHAVer) [9]. However, multi-domain model-based design and verification approach may cause non-negligible efforts to users because all aspect of the heterogenous models must be correctly connected. Furthermore, it is difficult to reduce the verification complexity by the model abstraction as maintaining these consistencies because if the model is modified at one side for verification complexity reduction the models on the other side also has to be modified to maintain the consistency. In order to achieve the efficient verification process complying with MBD, our approach automatically generates system models by integration according to only signal information between controller models and plant models [15]. Moreover, it can reduce the verification complexity by flexible component replacement method.

VII. Conclusion

In this paper, we proposed a practical verification framework for safety-critical CPSs that leverages automated model transformations and simplifications. The developed model transformation method automatically transforms bond-graph plant models with algebraic loops into signal-flow models without algebraic loops to make them applicable for existing automated verification approaches. The model simplification method automatically simplifies the plant model by replacing complex components by simpler ones that exhibit equivalent behavior to a sufficient degree. The replacement is based on expert knowledge, which is captured in replacement libraries for reusability, and application-specific parameter tuning based on simulation. We applied the proposed verification framework for the safety verification of a safety-critical automotive brake control CPS. The experimental results showed that the model simplification method yields approximately 35% computation load reduction in simulation and the model transformation method yields equivalent signal-flow models without recognizable errors. Additionally, the proposed framework was able to correctly detect unsafe behavior of the brake control system. In future work, we plan to develop an error localization method to aid debugging when safety violations are indicated by the presented verification approach.

Acknowledgment

Research supported in part by TUD CySEC. We also thank Hitachi Automotive Systems for providing the application examples.

References

[1] ACATECH (Ed.). Cyber-Physical Systems - Driving Force for Innovation in Mobility, Health, Energy and Production. 2011.

[2] Edward A. Lee and Sanjit A. Seshia. Introduction to Embedded Systems, A Cyber-Physical Systems Approach. http://LeeSeshia.org. ISBN 978-0-557-70857-4. 2011.

[3] The MathWorks Inc. Simulink R2015b. 2015. http://www.mathworks.com/products/simulink/

[4] Siemens PLM Software. Amesim version 13. 2014. http://www.plm.automation.siemens.com/en_us/products/lms/imagine-lab/amesim/

[5] The MathWorks Inc. Simscape R2015b. 2015. http:www.mathworks.com/products/simscape/

[6] The Modelica Association. The Modelica Specification, Version 3.3 Revision 1, July 2014. https://www.modelica.org/

[7] R. Alur. Formal Verification of Hybrid Systems. In Proceedings of the 11th International Conference on Embedded Software, 2011.

[8] T. A. Henzinger, P. Ho, and H. Wong-toi. HyTech: A model checker for hybrid systems. In Proceedings of the 9th International Conference on Computer Aided Verification, 1997.

[9] G. Frehse. PHAVer: Algorithmic Verification of Hybrid Systems past Hytech. In proceedings of the 8th International Workshop on Hybrid System: Computation and Control, 2008.

[10] A. Tiwari. HybridSAL Relational Abstracter. In proceedings of the 24th International Conference on Computer Aided Verification, 2012.

[11] A. Platzer and J. Quesel. KeYmaera: A Hybrid Theorem Prover for Hybrid Systems. In proceedings of the 4th International Joint Conference on Automated Reasoning, 2008.

[12] F. Lerda, J. Kapinski, H. Maka, E. M. Clarke, and B. H. Krogh. Model Checking In-The-Loop: Finding Counterexamples by Systematic Simulation. In proceedings of American Control Conference, 2008.

[13] H. Nakajima, S. Furukawa and Y. Ueda, Co-Analysis of SysML and Simulink Models for Cyber-Physical Systems Design, In proceedings of the 18th International Conference on Embedded and Real-Time Computing Systems and Application, 2012.

[14] R. Majumdar, I. Saha, K. C. Shashidhar, Z. Wang. CLSE: Closed-Loop Symbolic Execution. In Proceedings of the 4th International Symposium on NASA Formal Methods, 2012.

[15] T. Isihgooka, H. Saissi, T. Piper, S. Winter, and N. Suri. Practical Use of Formal Verification for Safety Critical Cyber-Physical Systems: A Case Study. In proceedings of the 2nd International Conference on Cyber-Physical Systems, Networks, and Applications, 2014.

[16] C. Cadar, D. Dunbar and D. Engler. KLEE: Unassisted and Automatic Generation High-Coverage Tests for Complex Systems Programs. In proceedings of the 8th USENIX Symposium on Operating Systems Design and Implementation, 2008.

[17] J. Magee and J. Kramer. Concurrency: State Models and Java Programing, Second Edition. Wiley, 2006.

[18] A. Bhave, B. Krogh, D. Garlan, B. Schmerl. Multi-domain Modeling of Cyber-Physical Systems Using Architectural Views. In proceedings of the 1st Analytic Virtual Integration of Cyber-Physical Systems Workshop, 2010.

[19] A. Bhave, B. H. Krogh, D. Garlan, B. Schmerl. View Consistency in Architectures for Cyber-Physical Systems. In proceedings of the 2nd ACM/IEEE International Conference on Cyber-Physical Systems, 2011

[20] A. Rajhans, A. Bhave, S. Loos, B. H. Krogh, A. Platzer, D. Garlan. Using Parameters in Architectural Views to Support Heterogeneous Design and Verification. In proceedings of 50th IEEE Conference on Decision and Control and European Control Conference, 2011

[21] T. A. Henzinger. The theory of hybrid automata. In proceedings of 11th Annual IEEE Symposium on Logic in Computer Science, 1996.

[22] A. Bhave, D. Garlan, B. H. Krogh, A. Rajhans, B.Schmerl. Augmenting Software Architecture with Physical Components. In proceedings of the Embedded Real Time Software and Systems Conference, 2010