

VIRTUAL HILS : A Model-Based Control Software Validation Method

2011-01-1018

Published
04/12/2011Yasuhiro Ito
HitachiYasuo Sugure and Shigeru Oho
Hitachi LtdMasahiro Matsushita
Hitachi Advanced Digital, Inc

Copyright © 2011 SAE International

doi:[10.4271/2011-01-1018](https://doi.org/10.4271/2011-01-1018)

ABSTRACT

This paper describes Virtual Hardware In the Loop Simulation (VHILS), a new validation method for real-time control systems.

VHILS integrates multiple-technology domain simulators and uses no actual hardware components. Target software in binary code formats are executed with CoMET®, a virtual processor platform and physical layer signals are emulated with MATLAB®/Simulink®. Thus, VHILS can replace HILS, which is widely used for control software validation today.

The VHILS was applied to the development of adaptive cruise control system (ACCS), and driver maneuvering, vehicle dynamics, microcontroller operation and CAN communication were modeled. The data exchange between multi-domain simulation and communication modeling were identified to be the primary causes of longer computational time. By carefully designing the simulator interfaces and the CAN model, we built an entire ACCS simulation that showed accurate match with experimental results and yielded one third of the real environment's turnaround time.

The VHILS was also applied to real-time software code debugging as an alternative approach to hardware-based In-Circuit Emulators (ICEs). The virtual real-time debugger (VRD) helps software engineers to examine run-time behavior of control codes and investigate cause-and-

effect relationships of software defections by synchronizing information from multiple-domain components and displaying them together.

To speed up the VHILS operation, Virtual Validation (V²) Cloud System, a parallel execution framework, is introduced. It automatically allocates VHILS tasks to massive computer resources and executes a large number of validation test cases simultaneously by using the modern cloud computing technologies.

INTRODUCTION

Automotive control engineers have been looking for an entire system-level testing method for validating dependability of their products. It is because the component-level validation which is used in traditional development does not successfully discover the faults caused by interaction in the complex systems.

One of the current problems of entire system-level validation is that control engineers can use only HILS (Hardware In the Loop Simulator). HILS is special purpose simulator which consists of I/O interfaces and a high performance computer. Fig.1 shows an example of HILS based validation environment. HILS is connected to actual target ECU and reacts their signals and disguises the mechanical components behavior. In this partial simulation environment, the real-time control software programmed in the ECU is validated as it is executed in the real product.

However HILS was found very much useful for software testing and widely accepted today, it still has these issues;

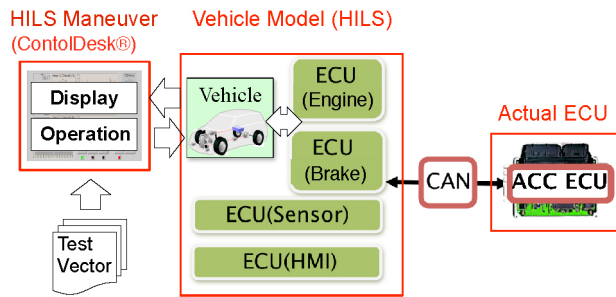


Figure 1. Control Software Validation using HILS.

1. Target limitation : As HILS is configured for a specific simulation target in order to operate real-time, it cannot be flexibly applied to other target systems.
2. Speed limitation to real time : HILS is connected to complicated electronic systems, and it consumes huge computer resources to maintain real-time response.
3. Visualization limitation : As conventional HILS needs actual microcontrollers, developers cannot observe the CPU related information such as state of software and internal interrupt signals.

With these reasons, validating system behavior with HILS will become inefficient when the automotive systems get more and more complex and multiple control functions are integrated into single ECU. For overcoming the problem of HILS, a validating method based on whole system simulation is proposed. It has been proven that simple embedded system consisting of one mechanical system interfaced to an electronic system such as engine control system is feasible in [1, 2].

In this paper, we propose VHILS, a model-based validation approach by integrating software, electric systems, mechanical systems and human maneuvering into models for multiple domain simulation. By focusing modeling method on the frequency of updating data and concentrating on emulating precise software behavior, VHILS can achieve simulation velocity as high as the real system. We also introduce the new validation technologies derived by VHILS. Those technologies realize faster and more effective entire system-level software validation.

MODEL-BASED VALIDATION SYSTEM FOR ACCS

The model-based validation approach improves the development of embedded systems. Simulation based validation approach has these advantages.

1. Flexibility : Developers can modify the target of validation simply by changing simulation model.

2. Speed : The simulation is accelerated as the performance of the computer improves, additionally, by splitting the process of test cases.

3. Full visibility : As all the functions in a target system are emulated on the simulators, functional responses are visible even the inside of LSI chips.

4. Variety of environment : By changing the parameters of a model which are fixed in real hardware, developers can validate software in various environment.

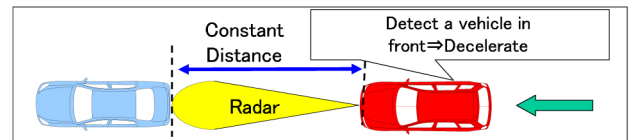


Figure 2. Adaptive Cruise Control System.

In order to realize VHILS, all the aspects of control systems need to be modeled using appropriate simulator.

VHILS approach is applied to development of Adaptive Cruise Control System (ACCS) which adjusts the vehicle speed automatically in order to maintain the safe distance against foregoing vehicles. (See Fig.2.) ACCS consists of a radar or a camera for measuring the distance, other vehicle control systems such as engine, brake or body, ACC controller which execute the software of ACCS and CAN(Controller Area Network) which connects them. Today, developers use HILS for vehicle systems simulation and plug an actual ACC ECU to validate its control software. Models for ACC ECU, CAN and human maneuver are needed for the whole vehicle simulation environment.

VHILS IMPLEMENTATION

Fig.3 shows the overview of VHILS, the virtual system for validating ACCS. The system consists of 3 main parts, automatic cruise controller model, vehicle and environment system model and the network model.

Adaptive cruise controller model runs on CoMET^{®1}, a CPU model and peripheral simulator. It consists of micro controller model, including CPU-core and peripherals such as timers, interrupt controllers and memories. They are modeled so carefully that they do not spoil the accuracy and velocity of the simulation. The behavior of these peripherals are analyzed in the respect of the frequency of affection to the software behavior, and omitted all the events which the software cannot recognize. The models are implemented with System-C or C language with libraries provided by the vendor of simulator.

Mechanical models run on MATLAB^{®2}, a physical domain simulator. The vehicle is exactly the same as the one used in the HILS. We developed Test Processor, which can operate

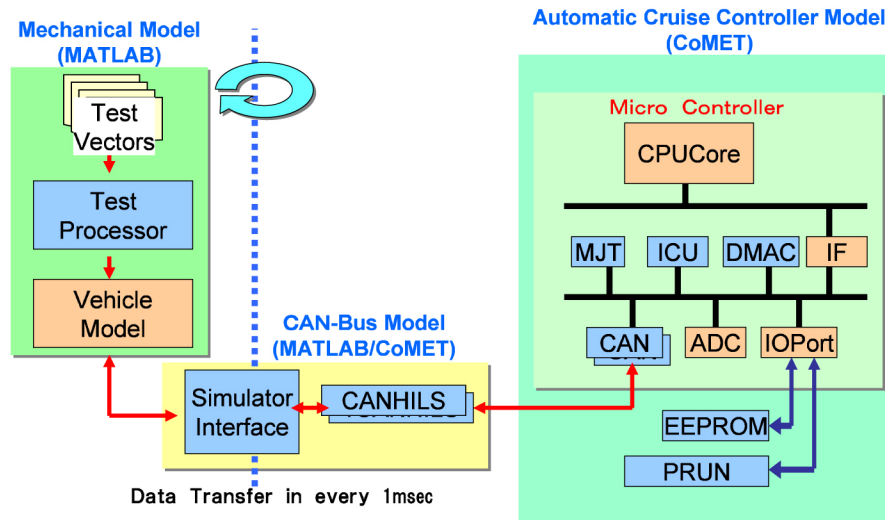


Figure 3. Structure of VHILS.

the signals inside the models and compose it to the vehicle model.

Network models are often major cause of deceleration of the simulation because of their excessive data-flow. To overcome this problem, the new modeling approaches were investigated for the CAN model. Virtual CAN model is integrated into the simulator interface between electrical and mechanical domain simulators.

TEST PROCESSOR

Generally HILS needs independent software for maneuvering test conditions operated inside the HILS model. That method cannot be used for maneuvering mechanical model in VHILS. It is because the software runs at only real-time because the target system of HILS are considered as running at real-time. However, execution speed of VHILS changes dynamically according to the conditions of computer which executes VHILS simulation. In order to synchronize VHILS models and normal HILS maneuvering software, they must observe the progress of the simulation each other quite frequently. The communication can be as frequent as 1000 times per one second, and it may affect the simulation performance due to the communication between them.

This problem is resolved by integrating mechanical models and signal maneuvering runs on separate software into one simulation. With this method, all the events are synchronized and adapted to the variable speed of simulation. However, building signal maneuvering function into VHILS model spoils the flexibility of test operation. While all the behavior of signal maneuver must be programmed before the simulations are launched, the signal maneuvering functions become complex structure with many branches and IO connection. Thus, signal maneuvering of VHILS implies two

contract issues ‘synchronization’ and ‘flexibility of test operations’.

```

/* This program means maintaining certain
   relative velocity with the preceding car */

void prec_vhcl_accl_spd
(double accl,double target_spd ,double wait_ms)
{
    double prec_spd;

    /* Waiting for the car model is ready */
    CALL( Wait_for_Time( wait_ms ));

    /* Waiting for the car model is ready */
    /* Use API function SetToOutput
       for setting signal value */
    SetToOutput( Radar_V1AccelerationLimit , accl );

    /* Get current relative velocity
       between the preceding car */
    /* Use API function GetFromInput
       for getting signal value */
    prec_spd = GetFromInput( RelativeVelocity );

    /* Waiting for the saturation
       of relative velocity */
    if( prec_vhcl_spd < target_spd )
        while( prec_vhcl_spd <= target_spd )
        {
            CALL( Wait_for_Time( 0.01 ));
            prec_spd = GetFromInput( RelativeVelocity );
        }
}

```

Listing 1: The sample code of Test Processor fragments

In order to satisfy both the requirements, the Test Processor is implemented as an automatically generated user-defined S-function program written in C language which runs on MATLAB®/Simulink®. Test Processor can maintain complete synchronization between models and maneuver by integrating signal maneuvering and its reaction inside mechanical model into one simulation. All the signal