Application of ISS-less technology to VirtualCRAMAS (SILS)

Yuu MORIYAMA
Takeshi FUKAZAWA
Masahiro MAEKAWA
Akira KITAMURA

Abstract

Conventionally, we have used HILS (Hardware In the Loop Simulator) for better inspection quality of vehicle control software in ECU (Electric Control Unit) before setting it up on an real vehicle. However, due to more complicated vehicle control systems and the division of development work, it has become difficult to obtain the ECU and build the HILS for the inspection.

There have been discussions on improving vehicle software inspection quality and/or efficiency, moving its inspection in ECU to an earlier stage than its actual setting, and verifying the logic in the upper process by adopting the SILS (Software In the Loop Simulator) that uses a virtual ECU instead of real ECU. However, SILS simulation speed is not fast enough and that has been an obstacle to practical use. [FUJITSU TEN TECHNICAL JOURNAL No. 47 in Japanese (No. 27 in English) carries the related article.]

In this paper, we explain our developed SILS "Virtual CRAMAS," which can inspect the software almost as accurately as the HILS does at faster speed. We developed it in the approach discussed here that simulates only microcomputer's operations necessary for the vehicle control software operation and using ISS-less technology, instead of adopting the ISS (Instruction Set Simulator) technology that faithfully simulates the microcomputer in instruction unit and was used for SILS introduced in the above-mentioned technical journal.

1 Introduction

In recent years, as vehicle control systems become more sophisticated and complicated, there are more ECUs (Electric Control Unit) in vehicles and, furthermore, multiple ECUs are integrated to control the vehicle systems. As for vehicle development, the trend of shorter development period and development without creating a prototype has accelerated. It leads to a serious challenge of how to develop the growing number of software effectively in a shorter time while ensuring quality of the software. Under these circumstances, SILS (Software In the Loop Simulator) is attracting the attention as a support tool for ECU software development without using hardware such as an actual ECU and HILS (Hardware In the Loop Simulator).

This time we developed "VirtualCRAMAS," which enables simulation at higher speed, in an approach that simulates only those microcomputers behaviors necessary for the vehicle control software behavior using ISS (Instruction Set Simulator)-less technology, instead of adopting the ISS method that faithfully simulates the microcomputer in instruction unit.

2 Types and Use of SILS

This section describes various types of SILS and their use.

Generally, in addition to HILS and SILS, the words "MILS (Model In the Loop Simulator)" and "PILS (Processor In the Loop Simulator)" are also used to refer to simulators. However, definitions and use of those words are different according to the literature and they are not standardized. Therefore, "HISL" hereinafter is defined as the simulator that uses hardware in the feedback loop and "SILS" as the one that only uses software and does not use hardware.

2.1 Emulation and Simulation

There are many types of "SILS." We categorize SILS herein according to "the simulation level (accuracy)" of the ECU including the microcomputer.

First, we define the simulation level from the system structure of an actual ECU shown in **Fig. 1**. As shown in **Fig. 1**, in an ECU, software layered into OS (Operation System), PF (Platform) software, app. (application) software, etc. runs on the hardware including peripheral circuits. As mentioned earlier, SILS simulates the ECU only using software. Therefore, the approach for the simulation is not limited as long as the SILS fulfills the purpose and meets the intended use. As a result, a variety of SILS have been developed for a wide range of purposes.

In this paper, we use the word "simulation" many times. Only in the context of ECU, it includes the meaning of "emulation." "Emulation" means running a part of a system on another system without any change. Among the examples is running software for Windows on Linux and running software of a game machine on a PC without changing anything. The key point is "as it is" or "without changing anything."

An ECU in SILS is implemented by emulating the software layers above a certain layer and by simulating the other below layers. When designing an ECU of SILS, an engineer determines first which part of the ECU is implemented "as it is" and "without any change." The determination is closely connected to the purpose of the SILS or the element of how and why the user uses the ECU.

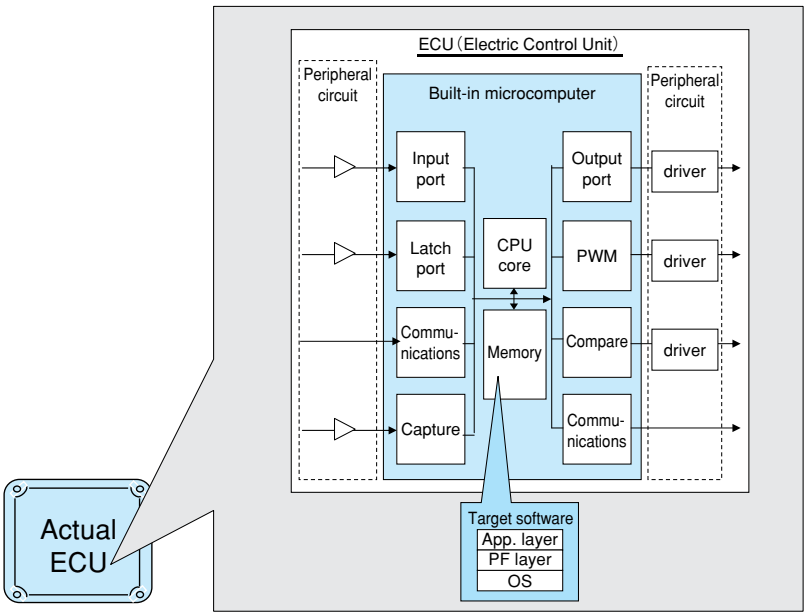

Fig.1 Structure of Real ECU

Obviously, the control software to be debugged by the user and the software upper than it is implemented "as it is" "without any changes." That means that it needs to be emulated. In this case, software lower than the control software (physical side, hardware side) can be implemented (simulated) freely within the range where the upper software is emulated.

Generally, the lower the software to be emulated becomes, the more precise and the closer to the actual equipment the simulation becomes (or the higher simulation level becomes). However, that requires high level of knowledge in software design and implementation and increases man-hours. On the other hand, the upper the software to be emulated becomes, the lower the simulation level becomes. However, such software can be easily designed and implemented.

The above-mentioned level is ultimately determined according to the use and purpose of the user. Whenever the simulation level is changed, the system becomes incompatible or cannot be reused.

Therefore, FUJITSU TEN developed an original method for realizing effective system development and user support by dividing SILS into three types according to simulation level and by using the same module structure and I/F for them. The following describes the three simulation types in details.

2.2 Three Types of SILS

Fig. 2 shows the three types of SILS, their use and characteristics that are defined by FUJITSU TEN. As shown in **Fig. 2**, according to the simulation level and use, those types are defined as

- **Type 1 (through RAM value)**
- **Type 2 (ISS-less)**
- **Type 3 (ISS)**

The Type 1 is the SILS that only covers "application software." Since it emulates the software upper than the RAM value area to which the application software refers, its structure is the most simple among those three and its operating speed is the fastest. We assume that it is used for system development or application development in ECU development.

The Type 2 is the one that covers "application software and PF software." However, it requires changes in a part of PF software. Since it emulates the software upper than the register area to which the PF software refers, it has more complicated structure and operates slower than the Type 1. We assume that it is used for application development in ECU development or development / inspection of PF.

The Type 3 is the one that covers "application software and PF software." Since it emulates the instruction set of the microcomputer, all software including the actual OS and PF software runs according to the object. As a result, its structure is the most complicated and its working speed is the slowest among those three. We assume that it is used for microcomputer development.

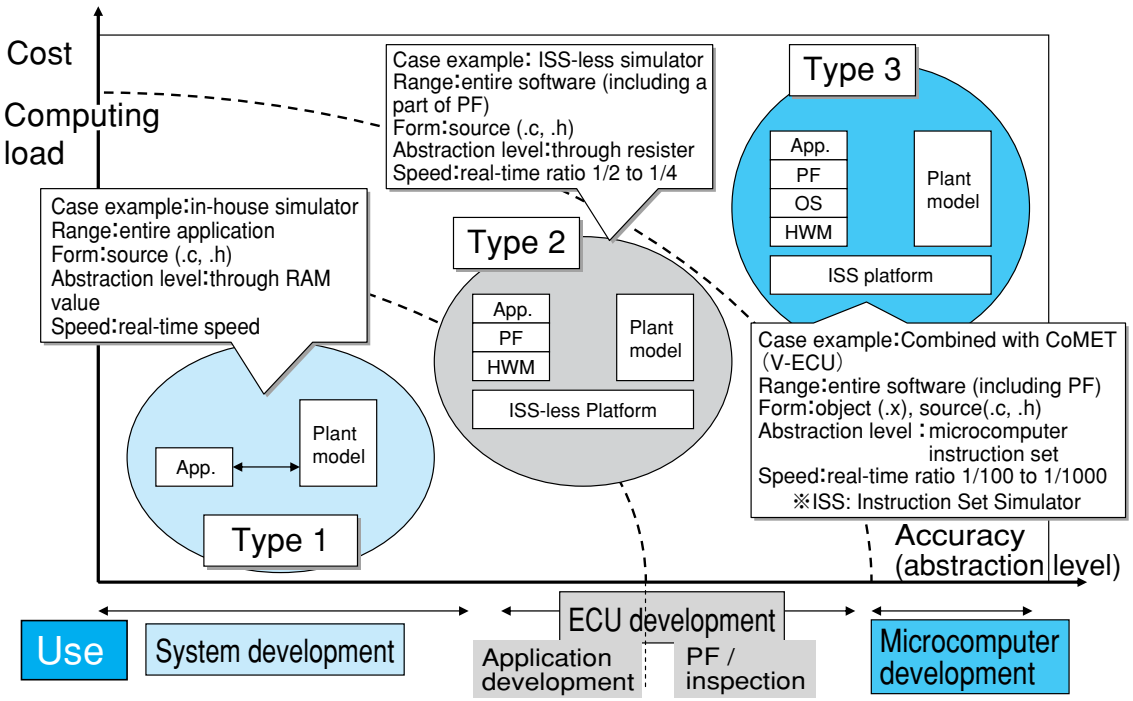

Fig.2 Types and Use of Our SILS

3 System Structure

3.1 Basic Structure of SILS

An appropriate SILS type is selected from the three according to the use of the simulation and their use is not limited by the type of intended control software. Consequently, the most appropriate type should be selected and two or more types should be used in combination even in the development of one model according to its development process and its purpose of debug. FUJITSU TEN divided the SILS, as shown in **Fig. 3**, into two parts: virtual ECU and virtual vehicle. That enabled us to realize the method where only the type of the virtual ECU can be changed easily according to the purpose.

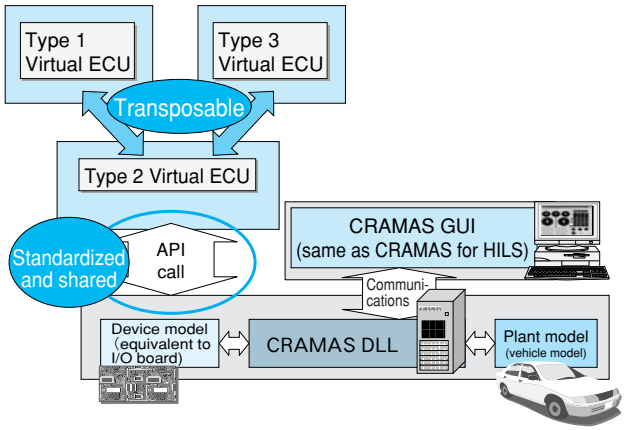

Fig.3 Basic Structure of SILS

As shown in **Fig. 3**, "the virtual ECU" and "the virtual vehicle" constitute the SILS. The virtual ECU can be switched from the Type 1 to the Type 3 and vice versa. On the other hand, a common vehicle simulator is used for the virtual vehicle part and is connected to the GUI that is used to operate the simulation and displays measured values.

That structure allows a user to debug the software continuously using the same vehicle model and test contents by switching the virtual ECU part according to the purpose. We made the GUI for setting and operation of the simulation to be shared by the SILS and HILS. As a result, the vehicle model and test contents for HILS can be also used for the SILS. Moreover, in the case of simulating the system in which multiple ECUs are integrated, it can optimize the simulation by adjusting simulation accuracy level for each ECU.

Our CRAMAS Division explained "the SILS Type 3" in our TECHINICAL JOURNAL No. 47 in Japanese (No. 27 in English). Therefore, in this paper, we elaborate the structure and internal algorithm of "the SILS Type 2."

3.2 Structure of SILS Type 2

Fig. 4 shows the detailed structure of the SILS Type 2. As shown in **Fig. 4**, the system consists of four main segments: "ECU simulator segment," "simulation engine segment," "external model executing segment" and "simulation control / display segment."

The ECU simulator segment is composed of "the virtual microcomputer core segment" with the control software and IO driver and "the virtual microcomputer resource segment" with the timer capture / compare function. The simulation engine segment consists of the event computing segment and the system clock.

The external model executing segment is constituted by "the plant model" that simulates external mechanism to be controlled, "the I/F board model" that simulates the I/F board function and "the model control segment" that controls those models.

The simulation control / display segment includes "the simulation control segment" that controls the entire simulation, "the data input segment" that inputs data to

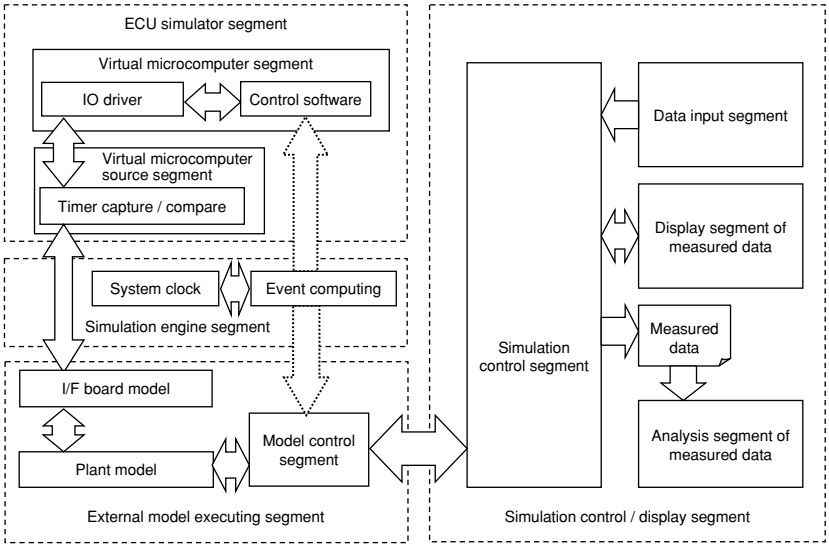

Fig.4 Structure of SILS Type 2

the simulating environment, and "the measured data display segment" that displays the measured results from the simulation. In addition, it has the "analysis segment of measured data" that analyzes the accumulated measured data in detail.

3.3 Application Example to Engine Control System

Fig. 5 shows the virtual structure of the SILS Type 2 corresponding to the actual engine control system.

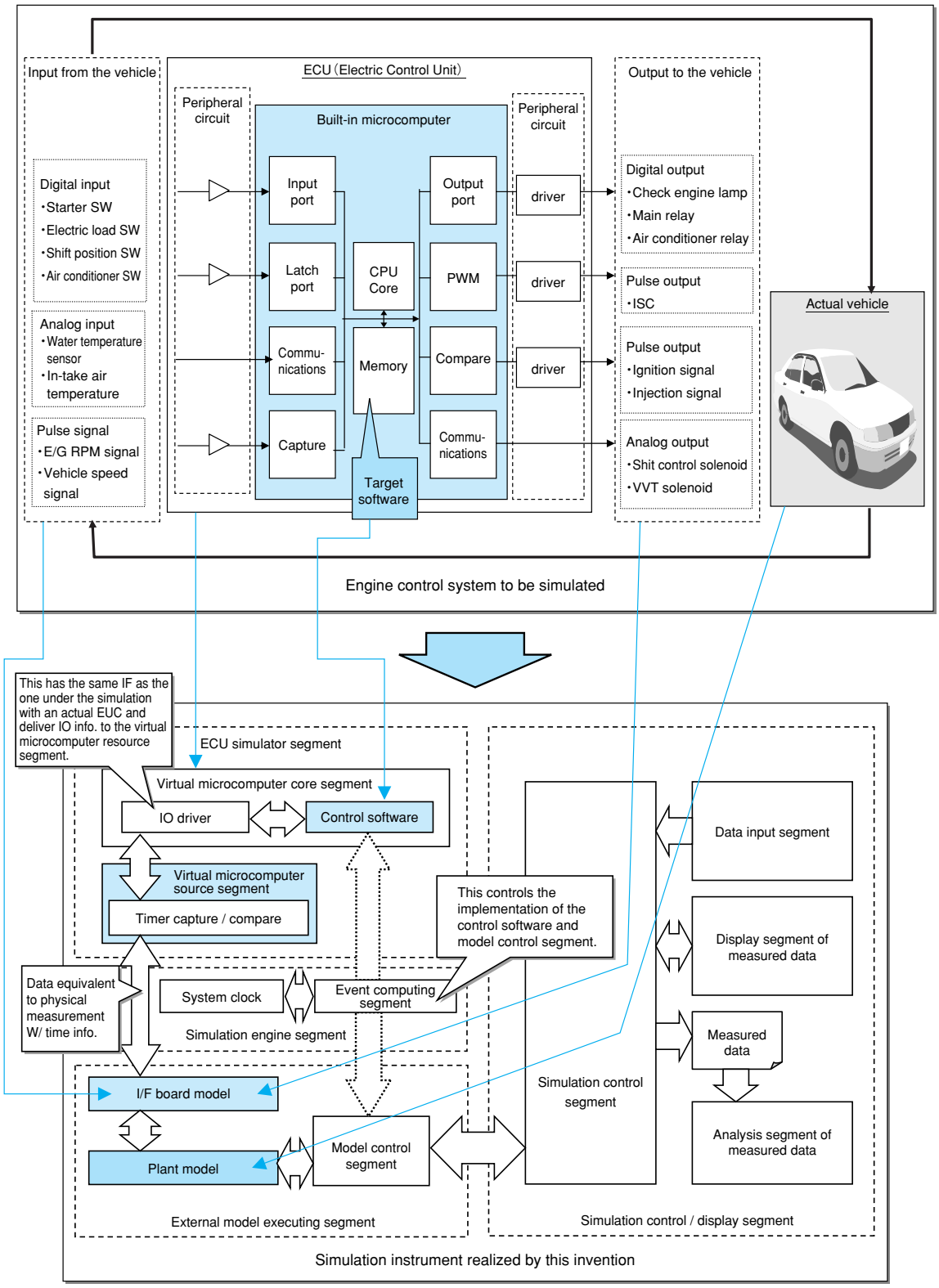

Fig.5 Structure of SILS Type 2 Corresponding to Real Engine Control System

As shown in **Fig. 5**, "the actual vehicle," "ECU" and "input to and output from the vehicle" in the actual engine control system correspond to the "the plant model," "the ECU simulator segment" and "the I/F board model" in the SILS Type 2.

In order to simulate the processing of the actual circuits on the microcomputer periphery, "the IO driver" is built in "the ECU simulator." "The simulation engine segment" is created between "the ECU simulator segment" and "the external model executing segment" to control the execution by "the control software" and "the model control segment." The SILS Type 2 can simulate faster than the SILS Type 3 because its execution is controlled by event processing, not by temporal synchronization processing. In the next and later sections, we will explain the method for the high-speed processing.

4 High-Speed Processing Method

4.1 IO Processing on Virtual Microcomputer Periphery

The SILS Type 2 realizes the IO processing on the virtual microcomputer periphery without hardware models such as a "CPU model" and "periphery circuit model" for high-speed processing.

Fig. 6 shows the method of IO processing on the virtual microcomputer periphery.

As shown in **Fig. 6**, "the virtual register" created between the virtual microcomputer core segment and virtual microcomputer source segment enables high-speed data input / output processing and interrupt processing needed to execute the target ECU control software.

We modified and added codes to a part of the hardware dependent layer and the microcomputer dependent layer for access from the control software to the virtual register for when data is input / output.

However, the modification and addition of those codes little affects the behavior logic of the control software so that it does not affect the behavior verification procedure and results of the control software.

And since their modification and addition can be defined automatically from the parameters or description rule of the control software, they can be automatically converted by the conversion tool.

Therefore, the SILS Type 2 can verify the behaviors of the application layer, the hardware independent layer, and the microcomputer dependent layer almost as accurately as an actual ECU and it can simulate faster than the SILS Type 3.

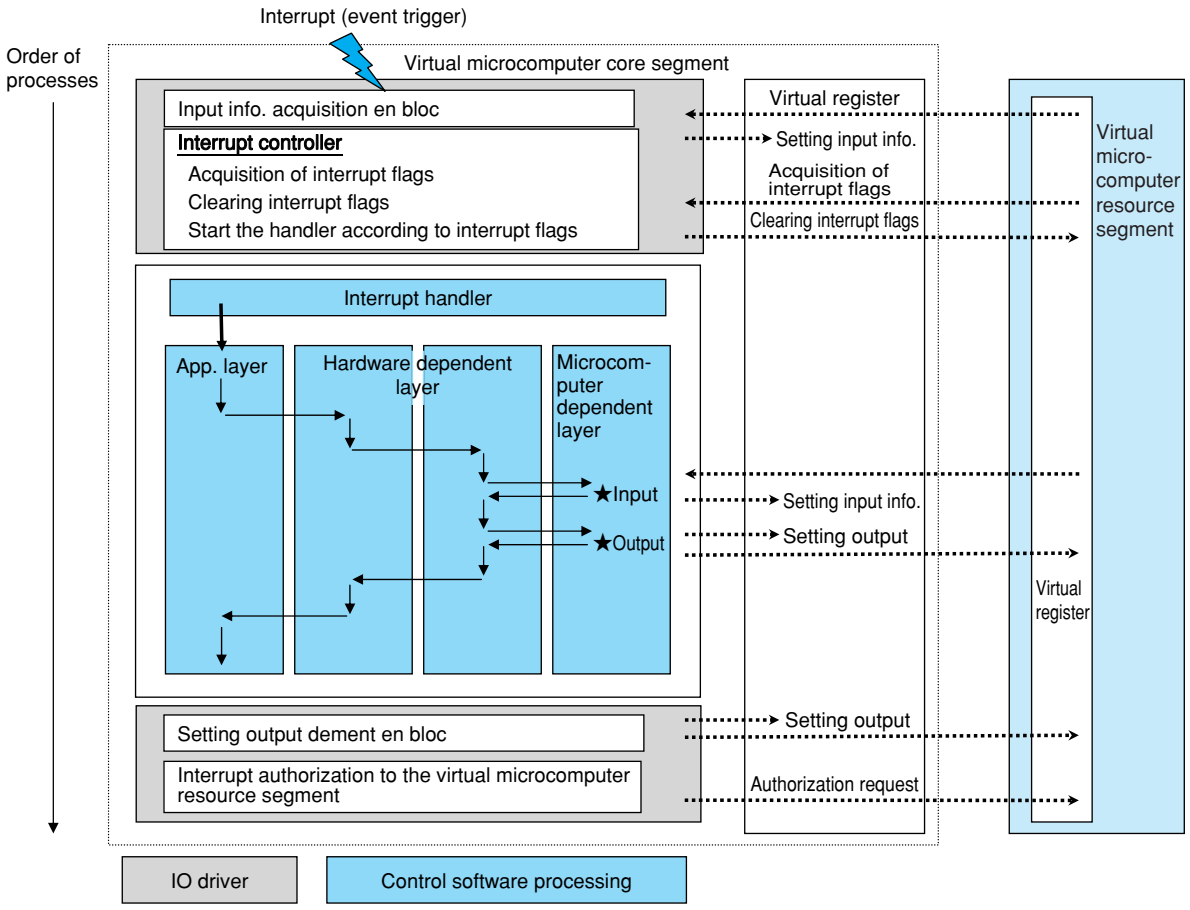

Fig.6 IO Processes on Virtual Microcomputer Periphery

4.2 Realization of Event-Driven High Speed Computing

In addition to the higher speed of the IO processing on the virtual microcomputer periphery using the virtual register, the SILS Type 2 enables the faster processing in the microcomputer by event-driving programming.

Fig. 7 shows the event-driven method for realizing the higher speed.

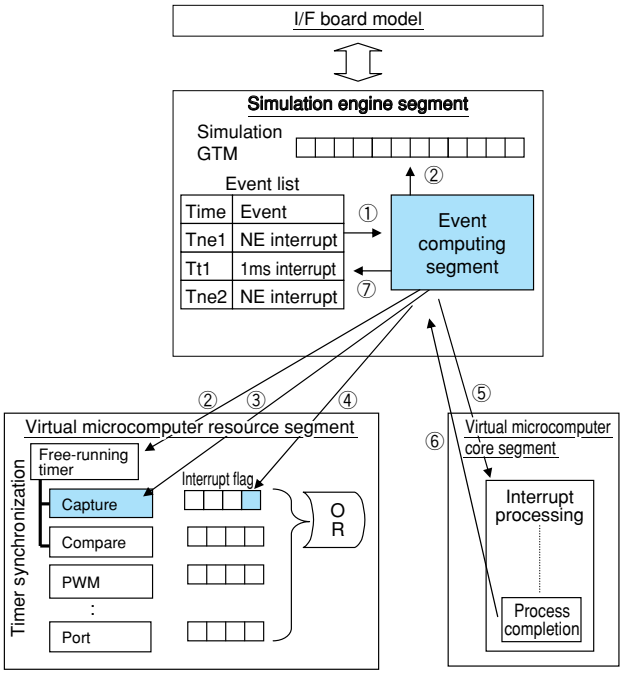

Fig.7 Method for Event-Driven High Speed Computing

The event driven procedures shown in **Fig. 7** are as follows.

①Take the topmost event out of the event list.

②Renew the simulation GTM (Global Timer) and free-running timer to time Tnel.

③Set nBit (capture register length) at the lower level of the free-running timer to the capture register.

④Set the applicable interrupt flag.

⑤Implement the calling equivalent to the interrupt of the virtual microcomputer core segment.

⑥Deliver the process completion.

The event computing segment calculates the time when the next interrupt occurs.

⑦Add the "Tne2" and "NE interrupt" to the end of the event list as the reservation of the next interrupt.

The series of the procedures enables higher speed operation than in the traditional routine computing method.

4.3 Time Concept in SILS Type 2

Fig. 8 shows timer processing at the event-driven simulation.

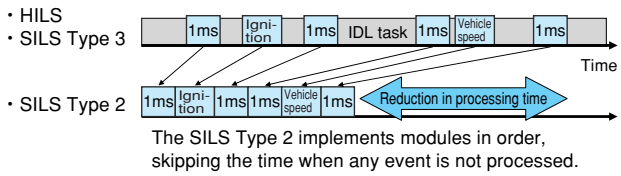

Fig.8 Event-Driven Process

As shown in **Fig. 8**, a timer (the upper in the **fig. 8**) ticks time at constant speed in the normal simulation. The only way to make the simulation faster was using a high-speed computer or reducing loads on processing other than on the target source.

On the other hand, in the case of event-driving programming, the constantly ticking timer is not required because events are processed in the order of occurrence. This method can slash the processing time.

Fig. 9 shows the relationship between actual time and virtual time of three simulation types.

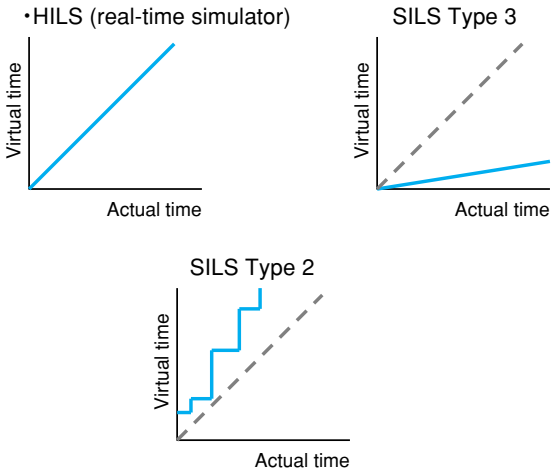

Fig.9 Relationship between Virtual Time and Real Time

In the case of the HILS (real-time simulator), the progress of the virtual time equals to the one of the actual time. As for the SILS Type 3, the progress speed of the virtual time is different from the one of the actual time. However, the difference is constant (in a straight line).

Contrarily, in the case of the SILS Type 2, the actual time does not progress (0 time) when the software processes but since the virtual time progresses by event processing, the time is skipped to a start time of processing. As a result, the difference between the virtual and actual time is not constant (non-linear).

5 Simulation Results of SILS Type 2

5.1 Input / Output Inspection of Virtual ECU Connector

In order to verify the SILS Type 2 behavior, we inspected the input / output ECU connector in the same way as we do for an actual ECU. As mentioned in Section 3, since VirtualCRAMAS can share the test contents with HILS, we used the same inspection sheet used for HILS. However, we did not inspect a part of signals related to electric power and other signals that were not modeled in the SILS.

Fig. 10 shows a part of the inspection results.

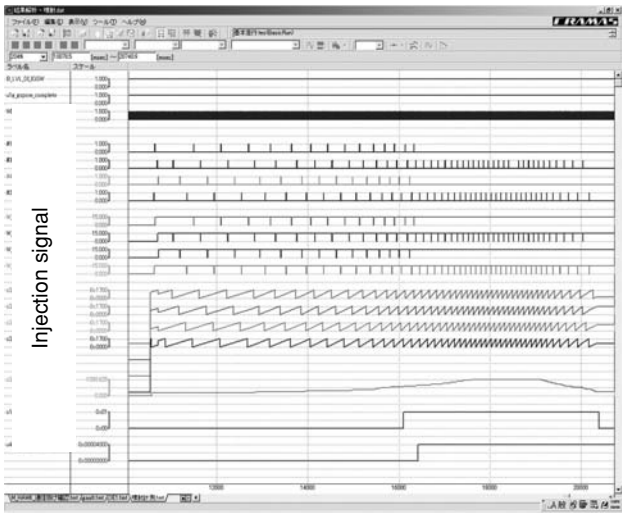

Fig.10 Inspection Results of ECU Connector Input and Output

We examined the results in **Fig. 10** in detail and found that all the signals satisfied the inspection criteria of an actual ECU. Most of the other signals also met them. We also found that the SILS modeling method or time concept was the reason why some signals failed to meet the criteria and that the ECU control software behavior did not have any problem.

In addition, **Fig. 11** shows the inspection results of when the same inspection was conducted on the SILS and the HILS.

The inspection results shown in **Fig. 11** indicate that the SILS's signal waveform and timing needed to verify software behavior coincide with the ones of the HILS.

As a result, we also found that the SILS Type 2 is usable for the inspection of input / output of an EUC connector.

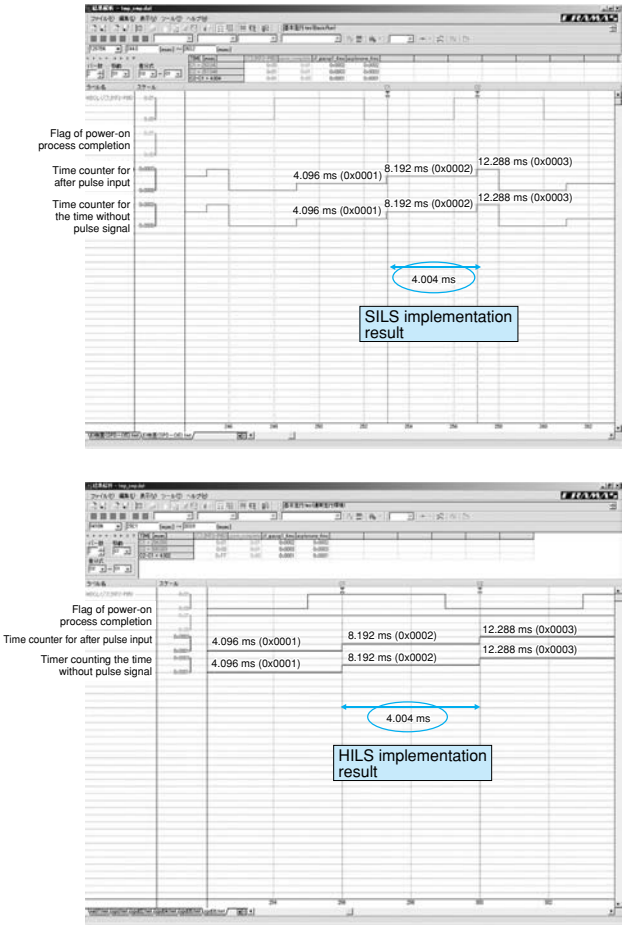

Fig.11 Results Comparison between SILS and HILS

5.2 Effects of High-Speed Processing Method

Table 1 shows the results of the benchmark comparison among our SILS Type 2 (VirtualCRAMAS using ISS-less technology) and competitors' SILS.

As shown in **Table 1**, VirtualCRAMAS using ISS-less technology, like the competitors' SILS Type 2, proved its simulation speed can be faster than the one of the SILS Type 3.

Table 1 Results of SILS Benchmark Comparison

	ISS-less VirtualCRAMAS	ISS-less SILS A	ISS SILS B	ISS SILS C	ISS SILS D
Type	Type 2 (ISS-less)	Type 2 (ISS-less)	Type 3 (ISS)	Type 3 (ISS)	Type 3 (ISS)
Manufacturer	FUJITSU TEN	Company A	Company B	Company C	Company D
ISS or ISS-less	ISS-less	ISS-less	ISS	ISS	ISS
Speed	100MIPS	100MIPS	1MIPS	20 to 50MIPS	30MIPS

6 Remaining Challenges

This time, by adopting the ISS-less method for VirtualCRAMAS, we developed the SILS so that inspection accuracy is almost the same as HILS and its operation speed is faster than HILS. Due to the higher speed, it can be applied in the development for the vehicle control software without an actual vehicle by using HILS and SILS, and it can streamline and improve the software development process. However, in this development, we realized the challenges and points to which we will need to pay attention when we put it into full practical use in the future.

6.1 Difference between Actual ECU and HILS

As mentioned earlier, we enabled the SILS Type 2 to simulate at higher speed by using the virtual register and the event-driven programming. However, this causes a difference in behaviors and time concept between the SILS Type 2 and a part of an actual ECU and / or HILS. As a result, the inspection results of the SILS Type 2 were occasionally different from when an equivalent test is conducted with HILS.

Therefore, when conducting an inspection with the SILS Type 2 based on the same inspection items used for an actual hardware or HILS, we need to note the items of which inspection results differ in the SILS Type 2 and an actual hardware / HILS. As a solution of this problem, for example, we create a common inspection sheet for SILS and HILS and specify inspection criteria of the applicable items for each simulator on the sheet. Defining the difference in advance leads to more efficient inspection because the same inspection can be conducted with SILS and HILS.

6.2 Reduction in System Building Man-Hours

When building the SILS Type 2, at present, we design and mount the HWM and the I/F board model in the virtual microcomputer resource according to each specification of the target microcomputer and peripheral circuits of the ECU.

Therefore, it takes many man-hours to build the whole system. We need to generate a hardware model library and examine the way to use the models in the library in combination according to specifications of a microcomputer and a peripheral circuit.

7 Conclusion

We developed higher-speed SILS by adopting the method using ISS-less technology for VirtualCRAMAS.

In the future, we need to make its simulation faster for the accelerated test and solve problems to put it to full practical use.

Lastly, we express our sincere appreciation to those who provided cooperation and help to us to develop VirtualCRAMAS.

Profiles of Writers

Yuu MORIYAMA

Entered the company in 1998. Since then, has engaged in the development of the simulator (CRAMAS) used for developing control systems. Currently, in the CRAMAS Department, Control System Development Division, AE Group.

Akira KITAMURA

Entered the company in 2004. Since then, has engaged in the development of in-vehicle electronic devices and development tools. Currently, in the CRAMAS Department, Control System Development Division, AE Group.

Takeshi FUKAZAWA

Entered the company in 1980. Since then, has engaged in the development of in-vehicle electronic devices and development tools. Currently, the Department General Manager of the CRAMAS Department, Control System Development Division, AE Group by way of sales engineering.

Masahiro MAEKAWA

Entered the company in 1990. Since then, has engaged in the development of in-vehicle electronic devices and development tools. Currently, the Team Leader of the CRAMAS Department, Control System Development Division, AE Group by way of vehicle control software development.