IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

Development and Testing of Electronic Architecture for Networked ECUs Using Diagnosis Fault System

Rahul B. Adsul[*1], Deepali A. Dekate[2]

[*1]AM, CIS Technologies (India) Pvt. Ltd., Pune, Maharashtra, India

[2]PVPIT, Department of Electronics & Telecommunication, University of Pune, Maharashtra, India

rahulbadsul@gmail.com

Abstract

The progressive increase in number of components and electronic system demands the overall architecture to be developed in different scenario .In order to achieve optimum electronic systems, it is necessary to build many real devices and evaluate the performance of systems. However, it is also becoming necessary to build virtual devices because of the increasingly complicated and large-scale systems. So the physical level, connecting between functional level and implementation level, should also be applied to virtual development. On the other hand, not only the functions, but also the safety designs need virtual technology to apply fault injection.

With increasingly sophisticated ECU development technologies, static simulators can no longer work with requisite testing requirements, so dynamic simulators are preferred. This progression with dynamic simulator will discusses the overall architecture of the system and the design decisions are made to reduce system cost.

This paper also discusses a concept and a powerful tool, which allows a wide range of automatic tests to be performed on networked ECUs. More precisely, it represents a complex system for connecting and testing all the networked ECUs in a modern vehicle.

Basically, testing is a vital and on-going part of the product development process, especially in the development of automotive systems. Validation testing of vehicle electrical systems and their computation is difficult and thus, expanding with the growth of certain features. Thus, a key to reduce test costs in increasingly complex systems is to work with the ability of the requisite distribution process in order to make every testable component in simpler manner.

Keywords: Electronic Architecture, ECU Development, ECU Modeling, Networked ECUs, ECU Testing.

Introduction

As part of the push towards a lower-carbon society, electronic control systems for automobiles are developing and evolving from domain-specific control in the vehicle (power train, body, safety, etc.) to the integrated control of the entire vehicle. The ECU, which forms the backbone of such control systems are thus growing in scale and complexity. The development of ECUs in this changing environment requires having an overview of the entire electronic system at the planning stages; this overview would set out an optimized ECU structure in which even the structure of the chipsets are defined; without such an overview, it will be difficult to keep up with vehicle requirements and specifications.

As well, the more stringent design requirements for safety that straddle multiple systems are becoming difficult to achieve using the conventional single-system, single-ECU development approach.

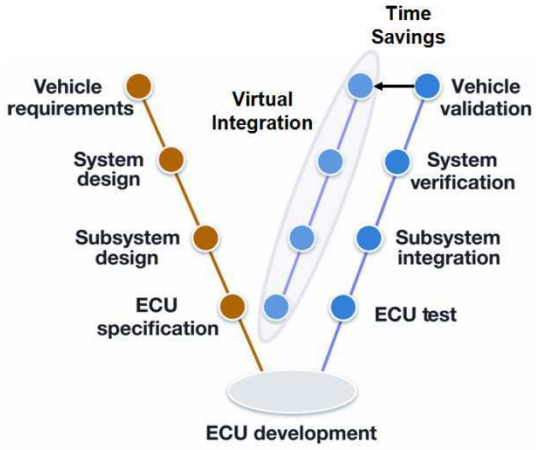

Figure 1: Process of ECU Development

Overview of Electronic System Development

Based on the requirements from the vehicle product plan, an automotive electronic system obtains vehicle information (values from various sensors) directly or through a vehicle LAN, analyzes it, and then cooperates with other systems or gives feedback to a particular actuator.

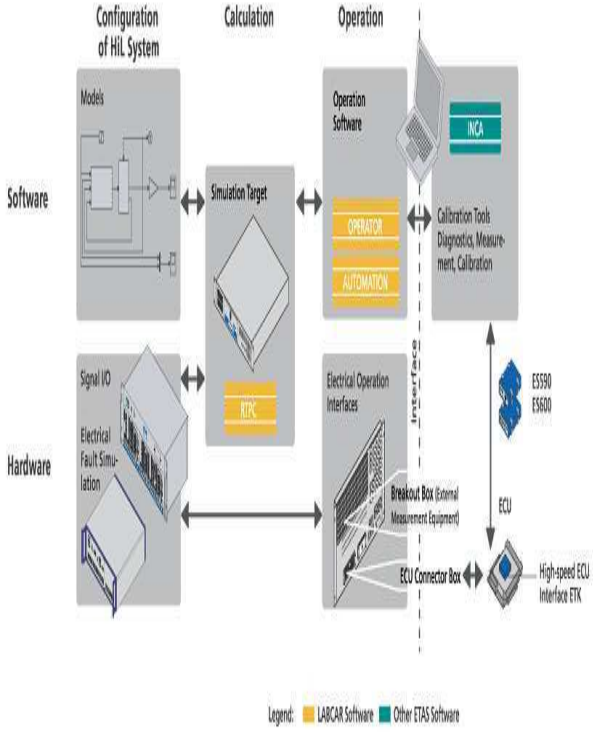

Figure 2: Configuration of Software and Hardware Units

Currently, to make the structure of the entire vehicle easier to understand, a block diagram simulators and other highly abstract theoretical models are used to develop architecture and to decide requirements for electronic systems.

When functions are being allocated among the various ECUs, the software and hardware structure of the ECUs is not considered.

In the following stage, the ECU development phase, the software and hardware allocation aspects of establishing system requirements is a major decision-making step. In this stage, the performance of the electronic system is mostly decided, and is followed by the concrete implementation design.

At the detailed design stage, the various constraints interact in complex ways. It is constraints in the system specifications and the constraints in the implementation. For example, a constraint in the system is to end processing within 1ms, a constraint in the implementation is to use a microcomputer.

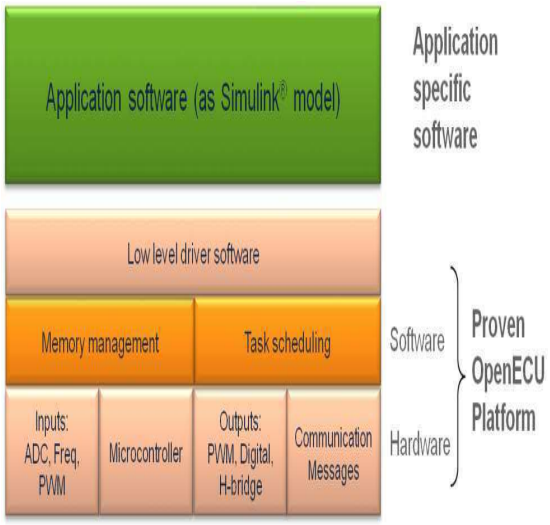

Figure 3: Linking Different Abstraction Levels.

If these are not balanced for cost, it is necessary to continue improving both specifications until it is balanced. This is because as the development process moves downstream, the amount of information required increases; however large amounts of information becomes apparent for the first time downstream.

In order to keep such coordination to the minimum, it is necessary to determine as much information at the upstream stages and to create a large-scale, detailed verification environment.

Aims of Virtual Development

Conventionally, optimizing systems and creating their architecture required fabricating large numbers of prototypes and evaluating them, but with their increasing scale, this method of optimization has become impractical. We therefore believe virtual manufacturing is a required step. As a function level simulator, a block diagram simulator is used, and as an implementation level simulator, SPICE is used today, these are both separate and independent development environments for the theoretical and implementation domains. Because of this, we will be working to introduce virtual development as a new physical level development environment to connect these domains.

In terms of not only function but also design safety, one of the elements that are required of virtual development is the ability to inject failures that are difficult to recreate in an actual machine. Thus it was decided to use modeling technology based on System C, a language that can be used to rapidly run system level simulations while having the notion of time, in order to create the virtual development environment.

This ECU modeling technology is described in more detail below.

Definitions of the Vehicle Electronic Architecture

We considered in the electronic architecture a specific module to control the functions of the engine, which is responsible for capturing the electric signals of the sensors management, and also, the ideal amount of fuel to be injected on the exact moment, through the time of opening and closing of the injection valves.

Another module is responsible for receiving the electronic signals of the foot pedal accelerator and also of other providing functions of the cabin, such as commands concerning engine brake, power take off, management of the sent or received information from the instrument cluster, and other ones.

Besides, these modules can also interact with other existent ECU's in the electronic architecture responsible to manage specific functions of the vehicle, such as: brakes, maintenance, gearbox and retarders, doors control, Immobilizer among others.

The following figures show two basic electronic architectures for commercial vehicles. The first of them displays a concept where the lines for diagnosis of faults and set of the parameters are made in an independent way, it means: each ECU possesses its own diagnosis line.

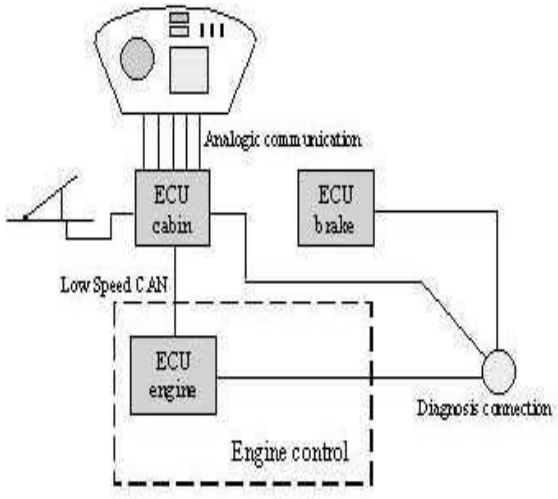

Figure 4: General ECU Architecture

In this kind of electronic architecture the Off-Board diagnosis equipment is responsible to address the messages for the ECU's through Instrument cluster diagnosis line. Therefore, the instrument cluster works like a "gateway for the diagnosis" receiving the information from Off-Board equipment and sending to the CAN BUS the communication data among the electronic modules.

Definitions of the Diagnosis Concepts

Currently, according to the country that the vehicle will be sold it is possible to use the SAE standard or ISO standard. Add to the protocols, are considered also the concepts to attend the maintenance of the vehicles in after sales. This is a very important point that should be analyzed because the conception of the electronic architecture happens before the procedures used to elaborate the diagnosis software that will be used in the workshops.

In this phase of the vehicle development is defined all the components and systems concerning to the On-Board and Off-Board diagnosis. Two topics will be examined:

On-Board Diagnosis – Nowadays, using the technological resources, a lot of applications for On-Board diagnosis can be improved in the commercial vehicles, considering the information showed to the driver are important for maintenance or detailed information about the vehicle functioning.

This screen can show all information that can turn better the vehicle conduction. A flexible service system can inform to the driver, through a display like this, the periods and kilometers values to change the normal wearing components of the truck, like: oil of the engine, oil of the gearbox and axles, brake components, filters, etc, depending of the conditions vehicle using.

Off-Board diagnosis – It is essential to make a very clear selection between the information that must be showed through of the On-Board and Off-Board Diagnosis. At the On-Board system, the fundamental objective is the possibility to the driver to see the necessary information to perform the best way to drive the vehicle and to discover the faults that is happening with the electronic system and power train.

In the Off-Board diagnosis system should be programmed all the necessary functions to do some changes in the ECU's and necessary information to perform a preventive and corrective maintenance of the vehicle in the workshop. Therefore, many diagnosis information require more technical knowledge.

The development, build, validation and release of software now require a strategy that must include global 24-hour coordination and test capability. More software has to be built, tested then released and customers have even greater expectations of quality than ever before.

Having tools that fit into this new high efficiency environment that are globally accessible and quickly configurable for different applications are essential for success. New requirements have emerged, including test repeatability, inter-related dynamic signals, higher number of I/O signals in new ECU designs, and the ability to configure a single simulator for multiple ECU programs.

This being the case, it is necessary to define an economical simulator solution that provided all of the features of the current static simulator including I/O and ease of use. This solution however would provide the advanced features that are now needed to develop and test ECU software in a corporation with distributed global engineering sites.

To get the most value from the simulator it needed to be on the bench and easy to use by a wide variety of users. The simulators design must be robust with high Mean Time Before Failure (MTBF) numbers and also must take into account normal use errors such as I/O wiring mistakes so all I/O need to be able to handle short to battery and short to ground conditions. Suitable diagnostic software also needs to be running in the simulator to alert the user to any issues that may exist in the system in order to avoid ECU testing errors. Careful attention must be taken to the design of the simulator's user interfaces, as technicians, hardware, systems and software engineers will all use the development bench simulator. Software testing requires the use of programming languages to create scripts and interfaces to other devices through Application Programming Interfaces (API's) especially when white box testing. This however should not be the only interface, as that would discourage other users from interacting with the tool. A Graphical User Interface (GUI) is therefore also required to make the system easy to use by the other engineering competencies.

The simulator, which uses a PC-based architecture to minimize costs, has specific I/O cards that can be easily reconfigured via software. Additionally, all simulator I/O cards were developed with a common FPGA communications core to reduce costs. This simulator also supports a wide array of commercially available PCI I/O cards for applications such as CAN, GPIB, IEEE1394, and other I/O needs.

For many years, the development of new vehicles has been characterized by the ever increasing use of electronic control units (ECUs). As legislation on environmental protection is repeatedly stiffened, e.g., CARB's OBD II standard, EOBD in Europe, mandatory reduction of fuel consumption, more and more complex engine controllers are required. Automatic gearboxes with new transmission concepts are also being increasingly used in medium-sized and compact cars. Electronic systems from the field of vehicle dynamics (ABS, ESP, ASR) are very often standard equipment in modern cars.

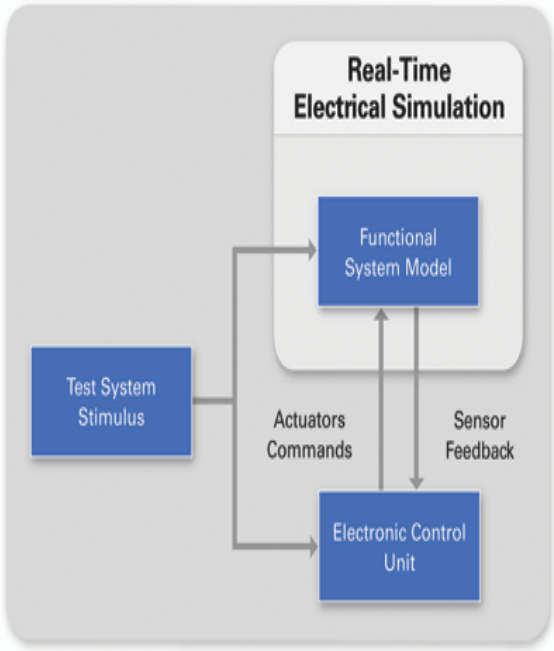

Figure 5: Real-Time Simulation.

Even for car body and convenience, ECUs have become indispensable. Thus many functions, e.g., seat movement, side view mirror movement, interior/exterior illumination, parking assistant and dashboard, are realized by means of ECUs. Implementing these complex functions is feasible only if the control units are interconnected via busses. This data bus networking of ECUs in the vehicle enables the sensor system, computed data, and the actuator system to be used jointly by a variety of functions. Typically, modern vehicle concepts consist of two or three different CAN networks. Particular ECUs, connected to more than one network, serve as gateways between the networks in these configurations.

The ECUs have CAN controllers (nodes) and are distributed on two CAN networks. The low-speed CAN network, the B-CAN, is connected to all body and comfort ECUs. The powertrain and vehicle dynamics ECUs are connected to the high-speed C-CAN bus. The body computer forms the gateway between the two CAN networks.

ECU manufacturers eliminate many errors during the project and development phases of the single ECUs. One of the standard tools, which have been widely used for years, is hardware-in-the-loop

technology. This particularly applies to all powertrain and vehicle dynamics ECUs. However, there are many other errors which cannot be detected without performing tests at integration and system level. This means that the complete system of networked ECUs must be tested.

Importance of Testing

Basically, the development of any product will require the verification of conformity to specifications and robustness in design. Testing allows the design engineer, test engineer or test technician to confirm that an ECU and/or system performs as intended. More specifically, it provides conformation that it can execute the functionality it was created to provide, and that it will successfully accomplish its task over its entire lifetime and through all conditions for which it was designed.

Now, the manufacture of a high-volume product demands uniformity in order to ensure first-run quality over the entire production run, from start to finish. Generating first-run quality of a production line eliminates the costly inefficiencies arising from reworking products that are not quite correct as they come off the line, and scrapping products that can not be reworked economically. Testing is included within the manufacturing process for use in establishing and maintaining uniformity control in production.

All development and production strategies basically, rely on testing for the feedback required to develop, produce and refine their products. Efficient development methodologies match testing scope and depth with the desired complexity and required robustness of the product in order to test the product optimally yet complete the test regime in a timely fashion. This process will maximize the test efficiency while minimizing the cost of the resources required.

Also, efficient production strategies seek to employ as little testing as possible, for efficiency, in order to support the uniformity that is their target. Historically, the most effective of these rely on statistical process control (SPC) as the means for managing uniformity in production. It is methodologies that detect the scope and depth of testing to be used.

Traditional Test Scenarios

Validating ECUs - The process of validating automotive ECUs generally involves exercising their functional capabilities while attempting to place them under controlled conditions that accurately represent those they will encounter in the target production vehicle. In this way the device under test can be scrutinized scientifically in its ‘natural’ environment.

For complex inputs and outputs like Exhaust Gas Oxygen (EGO) sensor inputs and fuel injector drivers, elaborate simulations of the corresponding production components are often used. However, to save money, test engineers have frequently attempted to use actual production components instead wherever possible. Invariably this sort of simplification results in a test that does not mimic the real world well, if for only one reason: it can not accurately represent the spectrum of variability encountered over the entire production run covering every part of the same design.

Take the case of the very simplest of I/O, the digital input. Responding to the state of a signal that has only two possible values, set at perhaps +12V and Ground, it would appear that the application of these discrete voltages by any power supply would be sufficient to represent the equivalent signals generated for the ECUs use by something elsewhere in the vehicle. However, an old axiom has it that every digital problem reduces to an analog one when problems begin to appear in the vehicle. In other words, even simple digital inputs have analog aspects that must be considered and accounted for.

Expanding this case to each input or output on each ECU in the vehicle highlights the degree to which test systems must be designed in order to avoid missing a failure. The only certain method for minimizing the needs for this level of detail is the reduction of I/O counts themselves.

Validating systems -Historically ECU validation and systems validation have been treated as one and the same in the automotive world. This is because each new system added to a vehicle has usually been built upon a single, and intimately related, ECU at its heart. Engineers have found that each new feature proposed is most easily developed as a separate ‘overlay’ to the existing vehicle electrical design. Thus each is generally assigned a new ECU and added incrementally to the existing array of electronic features already fitted to its vehicle. This concept has worked well until recently because these new and independent systems have been developed, refined and put into production as self-contained, standalone solutions having a minimum of interaction with the remainder of the vehicle’s systems. Quite some time ago a more formal adoption of the concept of systems began to be emphasized, focused primarily at ensuring thorough consideration of the effects of the external components making up each system, as well as interaction with the other systems in the vehicle, in addition to the ECU itself. Prior the time detailed study and characterization of these external components and effects was frequently forgotten or unconsciously minimized by ECU-focused engineers.

These oversights have frequently led to unfortunate results, primarily because no ECU operates in a vacuum in the increasingly complex vehicles that are being designed and built.

More recently, the systems approach has returned to prominence because a migration away from traditional discrete wiring is occurring. The migration first introduced, and then facilitated the expansion of, computer style networking in vehicles. This type of networking, originally called multiplex wiring, was first introduced as a means of reducing I/O, more specifically those inputs and outputs used as interconnects between ECUs. In practice it has resulted in a corresponding reduction in wiring cost, and so its use has been expanded dramatically over the last several years. Successful implementation of in-vehicle networking requires a systems focus because it raises the level of interdependency between ECUs on the vehicle.

As a result of the emergence and recent reinvigoration of the systems focus, and the clear need to test ECUs as part of the system to which they belong, more often than not, the validation of an ECU must go hand in hand with the validation of the system in which it resides.

Test Strategies

Of all design, development and manufacturing tasks, testing is perhaps most critical because of its ability to confirm the successful transfer of theory into practice. For this reason it is conducted periodically throughout the process of designing, releasing and manufacturing a vehicle from start to finish. During the early part of the design effort testing usually involves the simple confirmation that desired outcomes result when designs are run through their operating regimes. These early tests are very frequently ad hoc, informal, and not usually conducted according to a detailed time line.

The first formal testing event in most development programs occurs when the entire design is completed. At this point design verification (DV) tests are created and conducted to a detailed formal plan established prior to beginning the development process

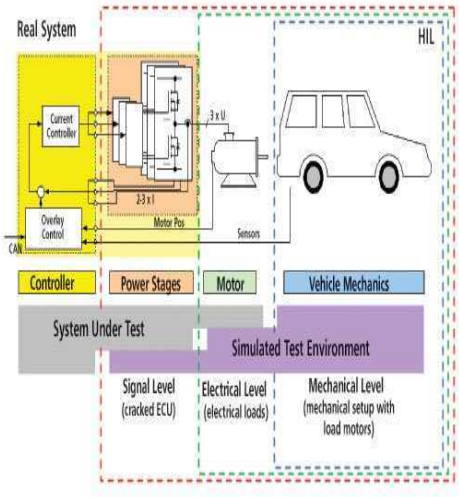

Figure 6: Vehicle Simulation Test Environment

The second formal testing event occurs with the startup of production. Process validation tests confirm the ability of the manufacturing process to meet its target production goals. This is essential to the establishment of a controlled production environment. Since many aspects of the design of ECUs and electrical/electronic components have an effect on manufacturability, this test set also provides feedback on the design process.

The process validation test suite is also important for the maintenance of the controlled production process after startup. In today's quality/cost-conscious environment, some production processes employ statistical process control as the means for managing the production process, and ensuring controlled production, in a cost-conscious fashion. The initial process validation suite is used to validate every ECU prior to and concurrent with startup. Afterward statistical process control allows it, or the relevant portions of it, to be applied to samples drawn at random from the production stream according to a pre-established plan, rather than testing every part. This reduces test expenses significantly, while simultaneously ensuring optimized quality.

Another formal testing event is known by the generic term 'End-of-Line' (EOL). End-of-Line tests are usually part of a 100% inspection program. By definition 100% inspection is at odds with the premise within statistical process control that only random samples of production output need to be tested to verify conformance to specification in a well-controlled production process. Thus, the existence of EOL testing is an admission that SPC is very difficult or even impossible to successfully carry out with some products.

Of the thousands of parts that make up a typical vehicle, experience has proven that ECUs exhibit this characteristic most often. As a result of the receipt of too many bad ECUs, i.e. those that made it past their respective production screening systems without being detected, OEMs frequently mandate EOL testing for most of the electronic components they buy. It is particularly true for complex ECUs that their inherent complexity makes it difficult for their manufacturing processes to hold all of their characteristics in control using SPC or by any other means.

Common types of Test Systems

The ad hoc testing that is conducted during the earliest stages of a development program is a special case in which test equipment is generally not considered a system but more of a collection of independent items, brought together temporarily by the design engineer to serve the purpose. Historically, little automation has been used. This reinforces the idea that ad hoc testing is not formal. Unfortunately, it also can lead to the incorrect assumption that it is not important as well.

DV testing is generally the first testing conducted within a development program to benefit from the construction and use of a formal test system. The main advantage of the introduction of formality is repeatability, ensuring consistent results that the engineer can trust. In recent years DV systems have come to be increasingly automated, in order to improve throughput as well as to ensure repeatability. PV testing is a manufacturing development process intended to examine the variability of the produced parts, and not necessarily the robustness of the basic design. Although most are developed independently, some PV tests systems are built directly upon the DV testers that immediately precede them in the development chain. Similar in construction to DV testers, PV testers feature additional capabilities necessary for tracking and comparing key characteristic and unit-to-unit parametric measurements statistically.

EOL test systems have many of the characteristics of PV systems, and in many instances are actually the PV systems themselves. The difference is primarily one of the perceptions. PV is the predominant term for production lines run by SPC, while EOL generally applies for lines run with 100% inspection of all parts produced.

The construction of dedicated automated test systems for use in ad hoc testing has generally been considered too expensive for widespread application. As with DV, PV and EOL testers, ad hoc testers are generally very expensive because of the custom requirements of the test suite. However it is also true that these requirements are sometimes less custom than in the test systems found further downstream. Next, over the years several attempts have been made at the design and marketing of generic automated test tools for ad hoc testing use, some better suited than others for this type of work.

Simulator Architecture and Design

The simulator was designed in a modular fashion to meet the above requirements. This modularity provides the user with several advantages.

- The user can configure the simulator to match the needs of the application under test. If additional resources are needed the user only has to add an additional I/O module.
- In the event that a module malfunctions the user can resolve the issue by replacing only the module that was affected, limiting down time.
- This modularity extends to the simulator's processor board, which is a standard PC, ATX motherboard.
- The simulator is controlled by a model running on a standard PC motherboard, which controls the simulator via the Base module. The Base Module is the communications hub that distributes the model's commands to all of the other modules in the chassis.

I n addition to being the communications hub for the simulator the Base module also performs power moding for the system.

This guarantees consistency of the rising and falling edges of the power-moded signals across all modules. All modules are update in each frame of the model and the changes are clocked in at the end of the model's frame.

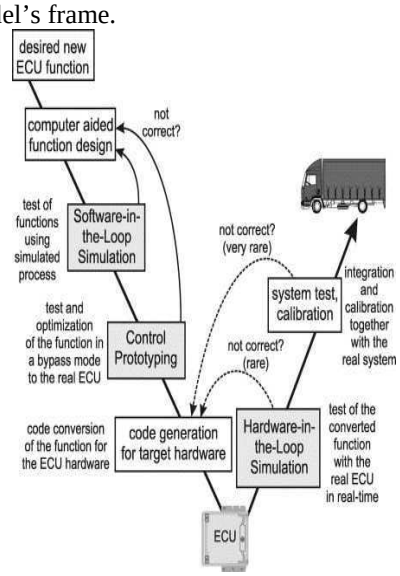

Figure 7: ECU Architecture Design

The simulator GUI provides a comprehensive, front-end for all simulator specific functionality as well as all adjacent enabling software: The simulators software, that runs on a Windows based host PC, is the interface that is used to view and control activity on the simulator. Using this interface the user can configure the I/O, select user defined display panels, assign I/O to the display panels, run scripts, and monitor and control simulator operation.

1. Configuring I/O - The user is able to configure the hardware and software settings for each module in the chassis via the GUI allowing each user or application can have its own configuration settings.
2. Display Panels - The panels used for monitoring and controlling the operation of the simulator are created using a commercially available graphical programming language software package. Every user can choose to design their own panels or they can reuse existing panels that were created for other applications. The software comes with generic panels that the user can use to get started.
3. Assigning I/O - All of the I/O for each module is presented to the user in a tree structure in the GUI. The user is able to assign any of these I/O to GUI control or display widgets on the panel enabling them to create customized interfaces. The user can then assign a name or alias to any of these I/O. This allows the user to see names that they will recognize and are common for like applications assigned to widgets on the display panel and in scripts.
4. Running Scripts - The GUI provides a control dialog for the creation, editing, running, and controlling of scripts. Scripts have the capability to set or monitor any of the I/O that is available on the simulator. Signals that have been aliased by the user can be accessed using their alias names. This means that the script references signals by the same names that are used on the display panels.
5. Monitor and Control - Once the simulator has been configured the user can monitor and control operation through the simulator GUI. In addition to the display panel widgets the user can use scripts, the Tactile Interface Module, or the API to monitor and control simulator operation.=

To minimize software cost per-unit, the simulator command and control station can utilize a scalable software set. For licensing purposes this software can be designated as either a development node, or a runtime node, or possibly an intermediate node. A development node would have the capability to edit models (using an appropriate modeling tool license), compile them, and create new GUI panels, in addition to all run-time capabilities. A run-time user would only have simulator configuration and control capabilities, without any additional software licensing costs. Under many deployment conditions, the number of runtime users greatly exceeds the number of development users.

I/O Configuration and Management

The simulator's configuration management process handles both the hardware and software configuration information for the system. Each user has the ability to create a configuration specific to their needs. The hardware configuration ensures that all of the modules needed for the applications are present in the chassis and that the correct cables are hooked up to each module. Due to the plug and play operation the order of the modules in the chassis are unimportant only their presence is important. These are used to determine that the correct cable is attached to the simulator and that the cable is hooked up to the correct modules for the device under test.

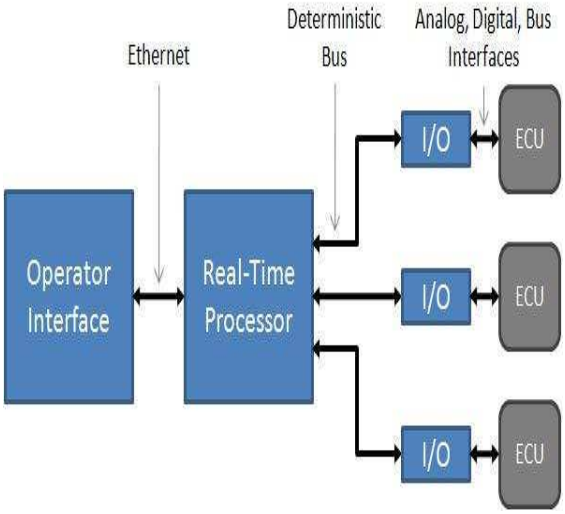

Figure 8: I/O Management

The software configuration ensures that the system has all of the files that are needed to boot the system. This includes all of the display panels, scripts, GUI configuration information and so on. The user has the ability to pack all this information into a single file for archive purposes or to send it to another user for sharing purposes.

Interfacing the simulators I/O modules into the chassis is a simple plug-and-play procedure. The simulator GUI automatically detects the type of card (if any) located in each I/O slot in the chassis during initialization. Through the GUI, an engineer can customize the signals range/scaling, voltage rails, channel configuration etc.

ECU Modeling Technology

ECU Modeling Concepts -

Before modeling, it is necessary to clearly set out how the virtual development is to be applied. First, in physical level design, it is important to determine how to structure the microcomputer, software, and peripheral LSI. Secondly, to validate whether safety requirement for ECU is satisfied, the following are required,

1. The optimized allocation of hardware and software.
2. Estimating the CPU processing load for the software to be installed.
3. Performing failure simulations.

Requirement 1 exists because of the need to devise a structure that satisfies the system requirements before the hardware or software even exists. Requirement 2 exists because of the need to have the software to be installed as well as a highly accurate CPU model. Requirement 3 exists because of the need for a way to inject failures and to perform simulations on the system as a whole. Though the models that would be used to meet requirements 1 to 3 all need different levels of abstraction(1), we believe that it is possible to come close to connecting these models with differing levels of abstraction in what is practically a single virtual environment.

Creating models of the interaction between ECUs, as well as the overall activity of the various sensors, control units (ECUs), and actuators being controlled will allow us to review the software and hardware structure and to calculate the CPU processing. Because the objective of the modeling is not only to recreate the overall behavior but to also have the notion of time, System C was used in all of the hardware models.

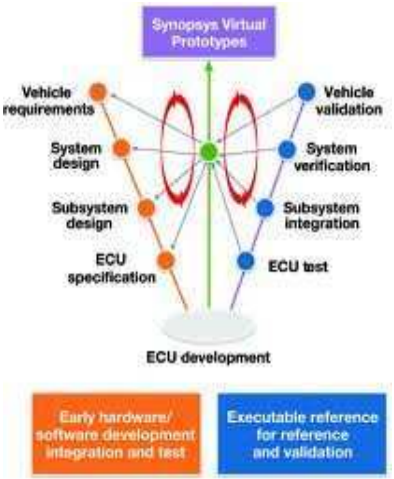

Figure 9: Process Flow of ECU Development.

Modeling Components -

Based on the existing system structure, we modeled each functional block including the AD converters in the ICs as well as the microcomputer peripherals such as the drive circuits and digital filters. This allows not only the overall activity to be observed but also the detailed behavior of each block. The microcomputer manufacturer provided the model of the microcomputer core, and this model was connected to the other models. The model of the microcomputer core is a cycle-accurate ISS model. By doing so, Requirement 1 (layout and review of hardware and software) and Requirement 2 (calculating the CPU processing load) are satisfied.

However, because having everything at a detailed level of abstraction results in the disadvantage of increased simulation time, the behavior in the models is investigated making strategic use (2) of transaction level and pin level interfaces between models to adjust the abstraction based on whether or not a block is under detailed review. This allows the total number of runtime events in the simulation to be decreased in order to create an environment in which large scale systems can be run at high speeds.

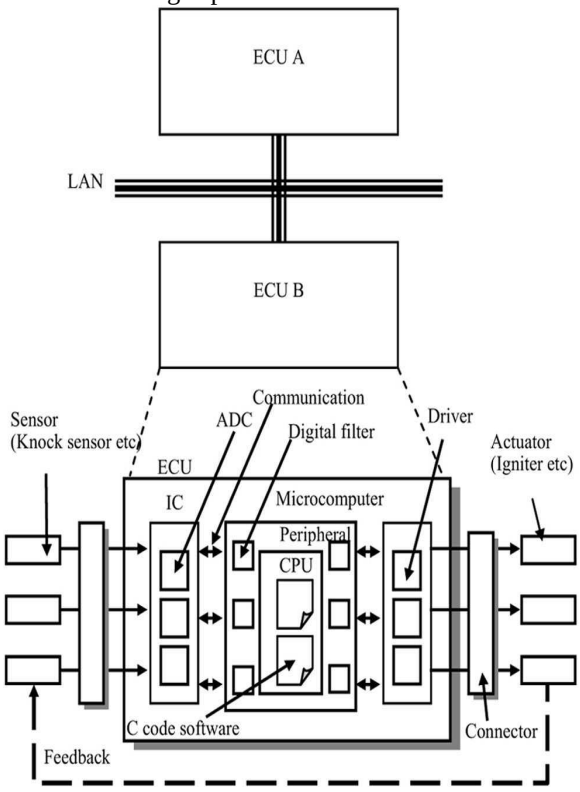

Figure 10: Interconnecting ECUs with Peripherals.

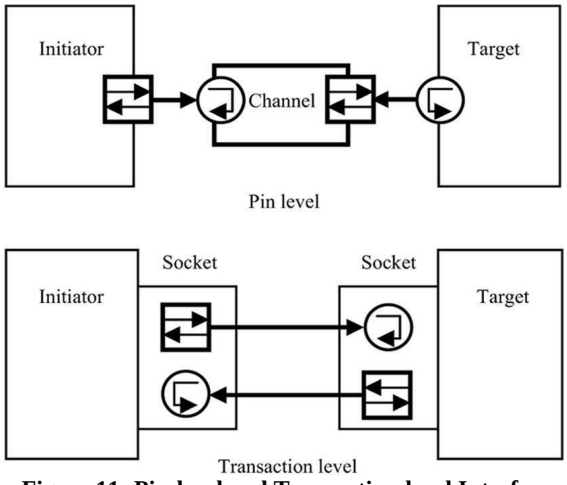

Figure 11: Pin-level and Transaction-level Interface.

When it considers how to model 32-bit communication line, transaction level interface is for verifying overall operation, and pin level interface is for verifying communication method. These are different abstractions. The degree of abstraction is frame-based for the first type and bit-based for the second type, and the simulation process has a single event for the first type and 32 events for the second type.

The disadvantage for the first type that bit errors during transmission cannot be simulated, and for the second is that the simulation takes too long. It was for these reasons that we developed a modeling method that maintained the advantages of both and resolved their disadvantages. The method is to add a switching event between transaction level and pin level modeling so as to enable dynamic switching. This reduced the overall simulation time, while still allowing performing detailed verifications.

Devising Failure Models -

Up until now, the modeling of functions has been discussed. However, there are advantages to a virtual development environment when designing safety into the system and when there is the need to ensure the completeness of fail safes for different types of failures. The operation of the entire system can be verified before manufacturing, and even after manufacturing, failures can be injected with accurate timing in the desired location without physically dismantling the system, resulting in the improved ability to control and observe failures.

To inject these failures, the failure modes were first analyzed. The results of this analysis revealed that failures can occur in various locations including physical connections and gates inside ICs, but all of these failures can be classified into a few modes such as disconnection and locking. Another issue is the location to inject these failures and how to inject these failures. Because the locations were failures can be observed in an actual machine are at its various terminals, a failure model was laid over the functional model as shown in Fig. 6, and the failures were defined in the output (a GND short failure is shown), which forces the system to treat the data transferred as abnormal values; this simplifies the failure model and makes failure injection easier, all without making any changes to the functional model.

The final issue is the timing of the failure. The failure model added above was given a failure changeover signal as an input with the value and time of occurrence set in the initial settings;

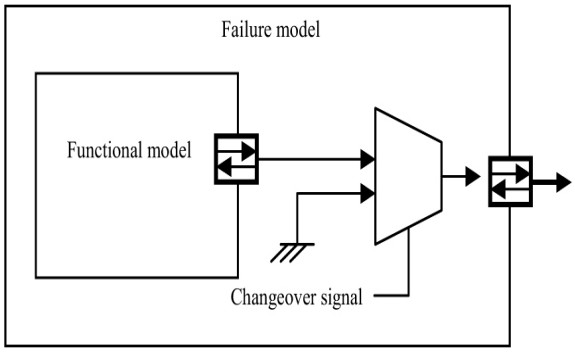

Figure 12: Failure model.

because such failures can be analyzed in the same way as a regular simulation, it is easy to express not only steady-state failures but also transient failures, and we were able to use this method to verify design safety.

ECU'S Development: Functionality and Diagnosis

For the ECU development should be considered the communication system that it will be used in the electronic architecture of the vehicle. In this way, the new ECU's should be based on common platforms to perform the data communication and the interface with another ECU's. A reference if this concept is the OSEK system (Open systems and their interfaces for electronics in the vehicle).

Many well-known European vehicle manufacturers and suppliers have joined up forming an international consortium under the name 'OSEK', which aims to introduce extensive standardization of software components. This mainly involves administrative mechanisms that create an environment for the actual application software to run in. These administrative functions fulfill practically identical tasks that are fundamental and consequently valid for all systems.

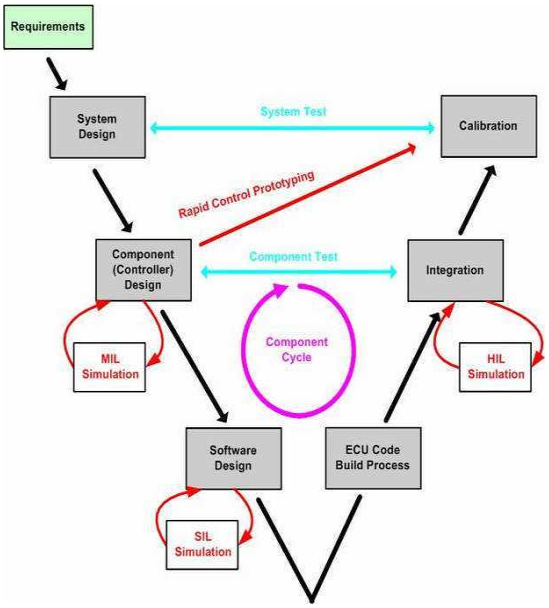

Figure 13: ECU Functionality

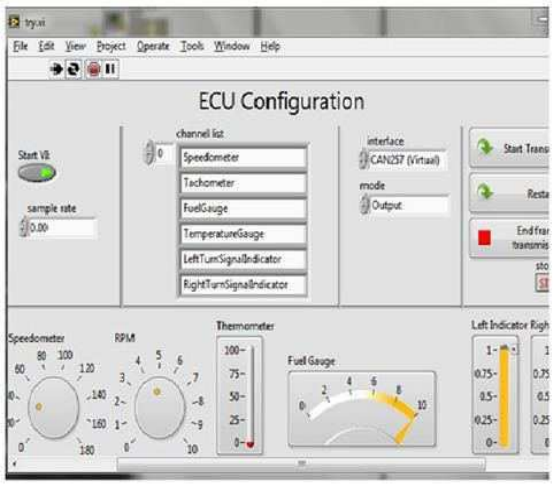

Figure 14: ECU Configuration

This operating system is a real-time multi-tasking operating system that is specially adapted to the conditions in motor vehicles. Particular emphasis is placed on the low memory and computing time required.

The OSEK communications unit governs the data exchange within a control unit and between different control units, regardless of the organization of the data bus system used. The network management sets the application bus system in operation, and permanently monitors all control units in the network.

The implementation of OSEK offers the following advantages when compared to the traditional development methods for control unit software,

• Improves quality: standardizing the basic software leads to higher quality, as modules that we have already been tested are used.

• Easier maintenance: the high level of modularity makes software maintenance easier.

• Re-usability: already tried-and-tested modules can be stored in libraries and used again.

• Easier to integrate: application modules, which are developed by different companies, can be linked together or available modules integrated into a new project more simply.

• Inter-changeable: modules form different companies with identical functions can be inter-changed.

Developing the Diagnosis Software

After the electronic and electrical architecture of the vehicle are defined by the development area, the after sales department starts the development of the diagnosis software (DAS) which will be used by the workshop network to fix the problems in the electronic systems built in the vehicle and also adapt them to the customer needs.

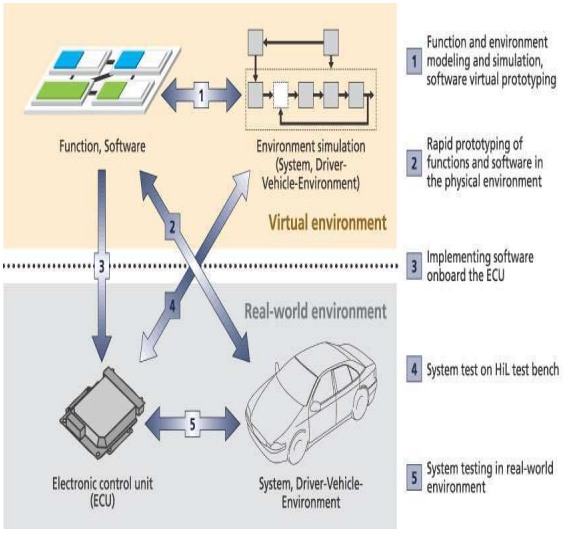

Figure 15: ECU in different work environment

The diagnosis software (DAS) is based on decision trees that provides a complete test and work procedure. Installation points and detailed wiring diagrams can be viewed during each procedure.

Realizing the Communication – It is necessary to be sure that it is possible realize the communication between DAS and the ECU.

To transmit the data from the ECU, the data from CAL layer has to be interpreted correctly. Creating single source files ensures it. The author has

the possibility to create it manually or through a specific application.

Creating single source documents – Single source documents are SGML documents "called" in decision trees (i.e. decision trees make reference to them) containing the following information:

• ECU parameters
• Fault codes
• Environment data
• Coding fragments

This application to create single source documents is available for all the authors via Intranet.

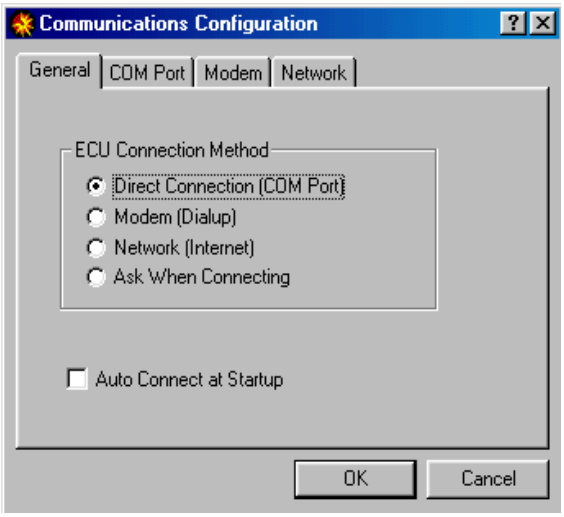

Figure 16: Communication Configuration window

Self Diagnosis Software

The ECUs are programmed with self-diagnosis software (on-board diagnosis), which allows management of the faults arising in the plant to be controlled (i.e. engine, transmission, etc.). In addition, the ECU passes the descriptive and standardized diagnosis trouble code (DTC) of the detected faults to diagnostic communication software via K-line or via CAN bus. Fault detection and the DTC management are specified by the European On Board Diagnosis (EOBD) standards in Europe and the California Air Resources Board (CARB, OBD II) in the US. These rules have been included and extended in the self-diagnosis specification (SDS).

DTC with the Basic OBD Structure

- Drive into specified operating point
- Activate electrical\logical\model fault
- Read out ECU diagnostic memory
- Evaluate test by comparing the detected fault with the expected fault
- Generate report automatically has been improved to include the EOBD test concepts.

Dynamic Software Verification

The focus of the simulation techniques covered to this point has treated the ECU as a black box – namely monitoring the I/O signals entering and exiting the ECU.

It is also often beneficial to monitor the state of measurement and calibration variables within the ECU during a simulation.

This provides the system with enhanced capabilities to perform validation, configuration control, calibration, and performance analysis including the following items.

- Setting an ECU input and checking the state in ECU memory
- Monitoring diagnostics and fault codes
- Uploading learn tables
- Adjusting calibrations
- Checking ECU software IDs for configuration control

Testing ECU Networks

Conventional Test Methods

Before the first vehicles prototypes are available, tests on ECUs (hardware and software) and other electrical components are performed on static benches that comprise the networked ECUs, the actual wiring harness and some of the sensors and actuators, i.e., the dashboard, the electrical motors of the seats, the control switches, etc.

These benches are normally used to test electrical actuators, simple sensors, and wiring harness, and to perform functional tests on the car body electronics and the self-diagnosis software. The network management, gateway functionality and CAN physical level are also tested. In order to perform these tests, breakout boxes are added to introduce electrical faults, power supplies are used to generate ground shift presence and some tools for CAN network and diagnostic lines analyses are used.

During vehicle development, the individual components are gradually replaced by prototypes that have previously undergone thorough testing. Tests are performed manually, so they are not fully reproducible and automatic test report generation is not possible.

The user's requirements for the test system are described below. They are fairly representative of other test systems for networked ECUs:

- All pertinent ECU power drivers and signal outputs must be read in by the test system. It must be possible to capture the signals and store them in files if required.

- The test system must be able to stimulate all the ECU inputs.
- Real electrical fault insertion capability is required on ECU outputs in order to verify how the system reacts to the insertion of known faults. For ECU inputs, electrical faults can often be stimulated by software.
- The test system must be able to log all CAN messages between the ECUs. To investigate the behavior of the CAN network, the test system must be able to perform the following tasks,
  1. Manage standard and extended identifier messages.
  2. Trace and record on all of the CAN lines simultaneously with time stamps.
  3. Send predefined messages interactively.
  4. Generate triggers on start of frame for detailed analysis.

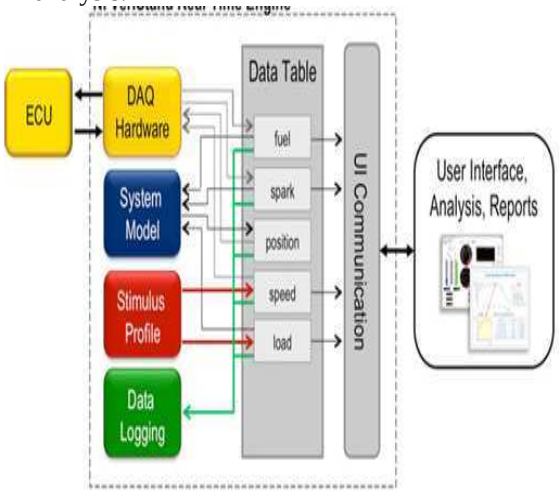

Figure 17: Data Communication with ECU.

  5. Measure the time elapsed between a certain message with identifier "x" and a message with identifier "y".
  6. Simulate the messages received and transmitted by nonexistent nodes and react to external triggers (events) or to events on CAN lines.
  7. Suppress all CAN messages sent by one or more ECU.
  8. Modify specific signals inside CAN messages and if necessary calculate a new checksum.
  9. Generate hardware errors on the CAN bus (e.g., by inserting additional capacitors or resistors between the CAN lines, generating error frames, destroying CAN messages at arbitrary bit positions).
- It must be possible to verify network management functionality: sleep mode, alive mode, and wake up mode.
- A diagnostic serial line is available on many of the ECUs constituting the test system. During test execution, it is necessary to interface the ECUs through this line to request diagnostic services and get diagnostic responses from the ECUs. In this particular case, the ability to interface to the ISO9141 serial line is required.
- Diagnostic communication protocols must be implemented based on this layer.
- From the ECU's point of view, the test system must behave like a real car. This requires real time capable models of all controlled systems, especially for the engine, transmission, vehicle dynamics and some of the body/comfort components.
- For manual interactive operation of the system, the experiment software must be powerful and flexible, but also easy to handle. The ability to automate the overall test system is crucial. For such a large system particularly, it is necessary to have powerful automation software with a well structured automation concept.

Components of the Typical Test System

Testing in its most basic terms consists of exercising the functionality of the device under test, conforming the conditions that are applied, measuring the resulting state of the ECU, comparing the results to a previous generated list of acceptable outcomes, and repeating this process for all functions.

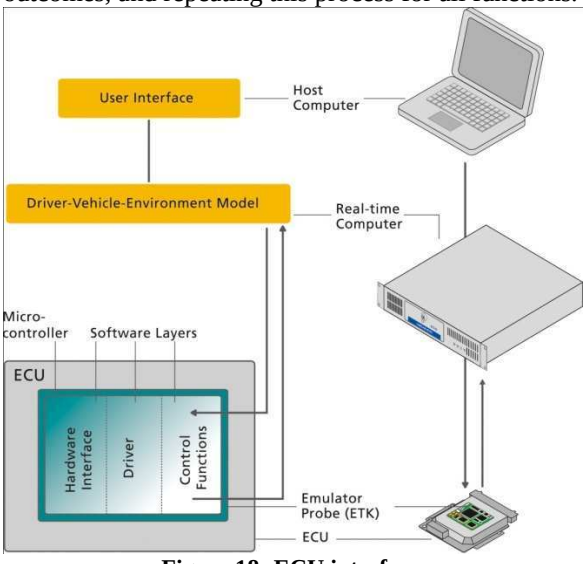

Figure 18: ECU interface

Thus a key part of the process is the application of representative inputs, and the presentation of representative outputs, to the ECU in order to place it in a controlled state that accurately depicts the real-world conditions it will encounter in

use. Test engineers usually refer to these inputs and outputs generically as 'loads', and their consideration, design and construction is a key part of developing any test system.

It is often tempting to use real sensors and actuators as loads. Following the line of thought that results in the acceptance of this logic, then it is logical to assume that test systems are simple to build and inherently accurate because they use real parts. However, in spite of being 'real', such parts can only substitute for one of a range of possible acceptable parts that could be used for the purpose. This is because all parts exhibit some variability. To be cost effective, ECU based systems are designed to accommodate this variability. In order to test most comprehensively this entire range must be traversed so as to ensure that all components in it, especially the ECU, can accommodate the variability.

A more accurate, and recent, method for presenting a representative environment to an ECU under test involves synthesizing, or simulating, the characteristics of all loads wherever possible. Simulated loads have more flexibility and test the device under test more thoroughly be4cause they can be programmed to simulate a part whose characteristics fall anywhere in the range of variability the system is designed to accommodate. The added flexibility that results is accompanied by substantial additional complexity setup and control. For this reason these loads are almost set up and controlled by a computer.

For this reason, along with several others, all modern test systems feature substantial computer controlled operation. Computers in testing are used to command loads to a particular configuration and also to command test instruments to conduct measurements. They are also ideal, however, for recording and analyzing the results, and particularly for having all of the above correctly with carefully controlled timing when called for.

Most importantly, while humans can generally accomplish all of these same tasks, computers have excelled at conducting them repeatedly, time after time, test after test, part after part. This consistency has become critical to the process of finding problems quickly and easily when parts fail a test, hence saving valuable time and money.

The last key part of a typical test system is the mechanism used to handle the connection of loads, measurement devices and power to the device under test. When an engineer sets up and runs an ad hoc test, all electrical connections required to conduct the specific test are usually made by hand, set up and broken down function by function as the test suite is run through its sequence. When a computer runs an identical test it needs a method for handling the electrical interconnects automatically, since it does not have hands with which to make and break connections.

In modern test systems this task is usually assigned to an electro-mechanical mechanism called a switch matrix. Switch matrices allow fixed and costly loads and measurements assets to be selectively connected, broken down and reconnected to the I/O pins of the device under test by computer control. The term matrix has been applied because the mechanism consists of an array of switching relays, each of which can be activated independently, or with others in concert, to establish the electrical path between the test equipment and the device under test. The matrix, under command of the computer running the test, is responsible for setting up and breaking down every connection needed during the execution of the test suite.

When comparing computer-controlled testing to that conducted by a human, the fact that the computer requires a switch matrix is the only nontrivial difference. While computers themselves are generally inexpensive and ubiquitous, giving them 'hands' in the form of switch matrices amounts to additional cost. Historically this cost has generally been accepted because the speed and consistency advantages provided by computer control usually outweigh the cost of the matrix.

Test Automation

Test automation via scripting is essential if ECU testing is to keep up with developer's needs. This task involves developing iterative, incremental test cases that can exercise an ECU through limitless test scenarios, and implementing those test cases in the supported scripting language.

To guarantee real-time performance, the system allows test automation scripts to be downloaded and run directly on the real-time processor in the simulators chassis. Through this mechanism, outputs from a running simulation can be channeled to data files for post processing.

Test Requirements

The previous items concerning the DTC have been formalized in a flowchart that describes the steps to be performed. The steps have to be performed for each DTC (i.e., P0201), for each fault symptom (e.g., short to GND, short to battery) and for each test condition (detection rules), e.g., power-on, cranking, engine run and vehicle run.

When a DTC, a fault symptom and a detection rule have been defined, a generic test could be performed as follows,

- Fault insertion with the correct fault symptom
- Check
- Fault off
- Check
- Fault insertion with different fault symptom
- Check
- Fault off
- Check.

Closed Loop Simulation

The simulators software suite contains a pre-compiled open-loop model, which can be reconfigured via parameters through the GUI. This open-loop model is essentially a signal-mapping tool, allowing the engineer to easily assign any physical signal to any virtual (or physical via the TIM) control or display.

Through the enabling software tools that are part of the simulators development suite it is possible to add a plant model to the simulation, and interface this plant with the physical I/O. This model integration is accomplished using Simulink, and the simulators software automates the compilation of the edited model into a real-time executable model.

Using the simulator software provides an optimal path for test engineers to migrate from open loop testing to closed-loop testing on the same bench as testing needs continue to grow in complexity.

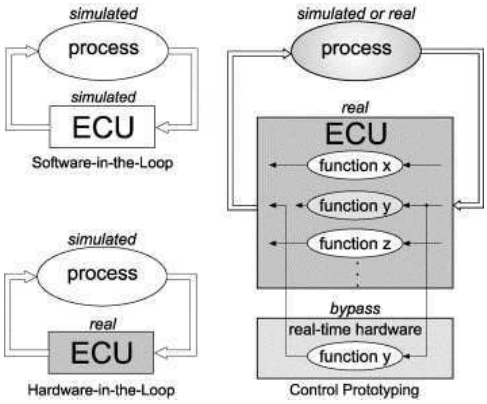

Figure 19: ECU – Closed Loop Simulation.

Another capability of the simulator software technology is distributed simulation. If the I/O count were to exceed the size of one chassis, their software automates the process of running the same simulation in parallel across two chassis. Similarly, if the compiled Simulink model were to grow in complexity to the point it could not be run at the desired step size, the model could be split into two smaller models and run on 2 processors in parallel.

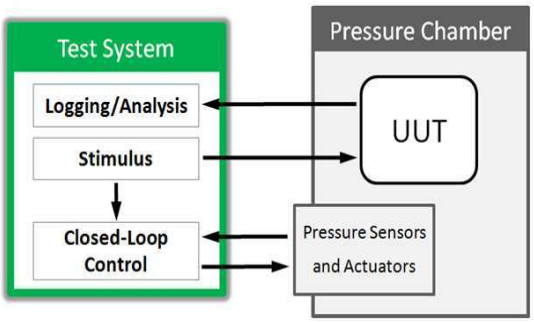

Figure 20: Closed Loop Simulation Process.

Through this software the simulator has the flexibility to scale itself to both increasing model complexity and I/O count easily and with minimal cost. Since all targets use the same real-time operating system (QNX), it is easy to interface the simulator with other simulation nodes in other form factors to further minimize costs.

Impact of In-Vehicle Networking

Control networks, Messaging and Signal transport – Automotive networks generally are of the command-and-control type. Unlike home or business local area networking (LAN), or location-to-location wide area networking (WAN), they are not typically used to carry large data files, e-mail messages, or internet content within the vehicle. Instead, they handle small data transfers called ‘messages’, which are typically used to convey commands from one place in the vehicle to another or to retrieve measurements taken by ECUs or sensors. They have evolved out of a concept called multiplex wiring, which is the basic technique for using a single set of wires to carry multiple signals.

The initial goal of multiplex wiring was the practical elimination of wires, interconnects, and harnesses through their replacement, where possible, by a vehicle data bus. This action altered the process of exchanging data between ECUs, between sensors and ECUs, or between ECUs and actuators. As a result, data exchange no longer consists of activating output pins and sending out digital or analog signals, but has evolved into a process of constructing messages containing representations of those signals and sending them out over the new vehicle network or data bus instead. However, as with networking elsewhere, substitution of a network for discrete wiring brings much more than cost savings on wiring. The data channel provided by the network supports many more opportunities for feature expansion, functional efficiency improvement, and systems cost savings than immediately meet the eye.

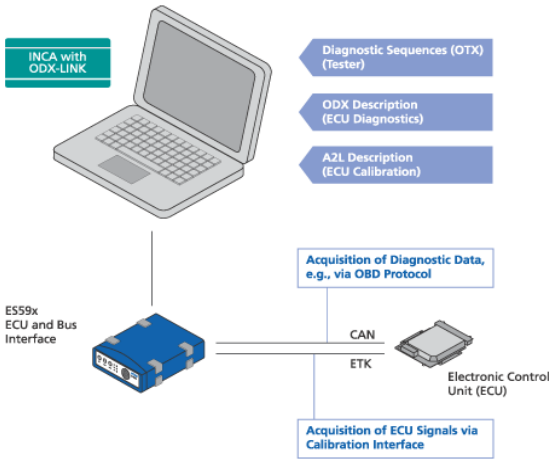

Figure 21: ECU Diagnostic and Calibration

Control networks are beginning to make a significant impact on the electrical systems of most vehicles. With the expansion of in-vehicle networking a major shift in the topology of the typical electrical system is underway, one that will have a profound impact on the test process, and ultimately shape the cost and complexity of testing.

Distributed I/O & Distributed functionality –

The concept of in-vehicle networking has evolved substantially since multiplex wiring was first introduced. However, it has generally followed a path that has emphasized a controlled and deliberate rollout of practical applications rather than an unchecked explosion of theoretical capabilities.

Sending signals over the bus eliminates dedicated I/O in a dramatic way as wires formerly dedicated to carrying them in traditional fashion are removed in favor of using the shared data bus. The bus has the effect of becoming a common pathway between devices, essentially allowing the small number of I/O dedicated to it to be reused over and over again.

The next major step was the introduction of the concept of distributed I/O. after implementing the first multiplex wiring systems engineers determined quickly that vehicle data buses could facilitate, at the system level, much more than simple communications between ECUs and components. They discovered that it has possible to use data buses to fundamentally alter the topology of the vehicle's electrical system, placing inputs physically near the sensors and input devices they connect to, and outputs adjacent to the actuators and output devices they drive.

The process of generating electrical systems with distributed I/O in this fashion is typically called geographical partitioning. I/O is moved from a centralized controller to a location in the vehicle much closer to the loads being queried or driven. This action has the effect of dramatically reducing wire harness length and makes the most sense currently in body and chassis control systems where functionality is exercised frequently across large areas of the vehicle.

At this point in time, given the introduction of more sophisticated data buses in vehicles, the concept is transitioning into the last major step in its evolution, distributed functionality. It expands the concepts behind distributed I/O by partitioning functionality as well as I/O geographically. This step involves widespread streamlining of existing vehicle functionality and the addition of new features and functions by splitting up the responsibility for these functions between ECUs.

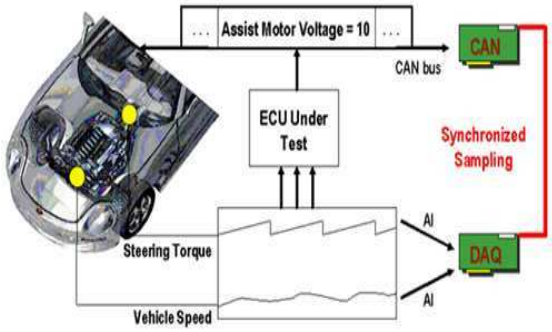

Figure 22: ECU Synchronisation

Taken to its extreme, this process facilitates the long-sought concept of smart sensors and smart actuators. Both feature traditional sensors and/or actuators fitted with at least some measure of functional control. Rather than placing this control in a separate ECU, smart devices split it up among themselves, reducing system complexity through the elimination of centralized ECUs. Distributed I/O is critical to accommodating the increasing numbers of I/O required to support the newest feature being added to vehicles, hence the concept of reducing test cost and complexity due to I/O expansion.

Distributed complexity is critical to reducing ECU complexity, leading to the reduction of test complexity and the elimination of non-value added costs resulting from what is essentially over testing.

Reduction of cost complexity and cost:

Since the costs are a significant percentage of the cost of developing and producing electrical systems, the reduction in test-costs that occurs from transition to distributed systems is worth considering in detail. The key to reducing test costs in increasingly complex systems is to exploit the ability of the distribution process to make each of the testable components in the system simpler.

Simpler systems are easier to test because they have fewer characteristics to monitor. While the total number of characteristics for a given system will

not change substantially in the transition from fully centralized to full distributed, there will be significant reassignment from ECUs to smart components, and from discrete I/O to signals embedded in messages sent across the network. This phenomenon will result in a reduction of testing complexity and cost overall for two reasons,

1. **Cost of testing to support quality –**

The inherent complexity of most ECUs, especially those with high I/O counts, makes it difficult if not impossible to successfully implement SPC in their production processes. This leaves costly 100% inspection as the only means of guaranteeing quality in those processes. By eliminating complex ECUs the simpler components that remain become excellent candidates for SPC and its inherent cost savings, because there are fewer key characteristics to monitor for each one.

2. **Core Cost of Automated Testing –**

Systems that use an In-Vehicle Network extensively has fewer traditional I/O, in many cases dramatically fewer. These systems are much less costly to test than those interfacing via many individual wires, because of the opportunity to use smaller, less costly switch matrices instead of larger, more expensive ones. The cost of switch matrices quadruples (approximately) for each input added, because each new input must be switched across all outputs, not just one corresponding new one. Additional costs not related to testing are also saved, for instance the cost of the complex I/O drivers inside ECUs used to handle the interface to individual wires, but these are outside the realm of testing.

Conclusion

Here, we believe that maximizing the performance of ECUs in electronic systems, which continue to grow in scale and complexity, and ensuring that these systems meet design safety requirements will require methods to visualize things that are difficult to visualize, and that this visualization is needed both before and after manufacturing. We would like to use the modeling technology described in this paper as a base for creating a virtual development environment and to carry out the development of vehicle electronic systems and products that contribute to society.

The simulation process provides the capabilities to replace the static simulator used for automotive ECU development and provides the advanced features desired by development community.

Many new ways of developing and testing ECUs and their functionality are discussed. These will provide a reduction in test execution time, reliability of tests due to repeatability of internal and external conditions. This will also provide the ability to perform more exhaustive tests by modifying the test conditions.

Also all the information related to diagnosis is available on the same software, which results in decreased run-time and other search efforts. This also results in decreased vehicle a downtime and service time that increases customer satisfaction.

Reduction in the cost of testing with increased distribution can be resulted in the necessary change which is helpful in many other ways to work with these distributed systems, along with the test benefits.

References

[1] Keyword Protocol 2000 (Implementation of Diagnostic Services - Recommended Practice - Version 1.1 - 31.01.1997).

[2] Ralf Pfaff - Diagnosis via CAN BUS On-board / Offboard - Version 1.13;

[3] Ralf Pfaff - Diagnosis on K-line (On-board / Offboard) - Version 1.1

[4] Luciano Breve Abrahão et al - Navigation Systems in Brazil - Needs and Requirements for Implementation - SIMEA 99;

[5] STARC: TL Modeling Guide, Second edition, Japan, Semiconductor Technology Academic Research Center, 2008, p.277.

[6] Köhl, S., Lemp, D., Plöger, M. , “Hardware-in-the-Loop Simulation” pp.948-955,ATZ , Wiesbaden, Germany, October 2003.

[7] Di Mare, G., Ferrara, F., Scala, S., Sepe, E., “Hardware In the Loop testing of EOBD strategies”, Proc. of the 15th IFAC World Congress, Barcelona, Spain, July 2002

[8] Caraceni, A.,De Cristofaro, F.,Di Lieto N., Ferrara, F., “Gasoline Rapid Control Prototyping System” 16v. Proc. of the AVEC ‘02 Congress, Hiroshima, Japan, September 2002

[9] Caraceni, A.,De Cristofaro F.,Ferrara, F.; Philipp, O.; Scala, S., ”Benefits of using a realtime engine model during engine ECU development”, Proc. of the SAE World Congress, Detroit, USA, March 2003

[10] Gruber, J, “ Steering Wheel Angle Sensor for Vehicle Dynamics Control Systems”, Proc. f the SAE World Congress, Detroit, USA, February 1997.