Proceedings of FUELCELL2006
The 4th International Conference on FUEL CELL SCIENCE, ENGINEERING and TECHNOLOGY
June 19-21, 2006, Irvine, CA

FUELCELL2006-97257

DEVELOPMENT OF AN AUTOMATED CONTROL SYSTEM VERIFICATION PLATFORM FOR A SOLID OXIDE FUEL CELL

John Absmeier
Delphi Corporation
Technical Center Rochester
Rochester, NY
john.absmeier@delphi.com

Tuhin Das
Emmeskay, Inc,
47119 Five Mile Rd.
Plymouth, MI, 48170
tuhin@emmeskay.com

Swaminathan Gopalswamy
Emmeskay, Inc,
47119 Five Mile Rd.
Plymouth, MI, 48170
swami@emmeskay.com

Ravi S. Paike
Emmeskay, Inc,
47119 Five Mile Rd.
Plymouth, MI, 48170
rspaike@emmeskay.com

ABSTRACT

The fuel cell industry is currently undergoing rapid development, and applications of fuel cell based power sources are diversifying. The advent of new and more sophisticated application areas and the expanding market necessitates development of efficient and robust fuel cell based power supplies that are reliable in their performance. These demands are answered not only by improved plant designs and innovations, but also by developing high-quality control algorithms. Quality and reliability of the complete system are ensured through extensive and varied testing. To this end an automated Hardware-in-the-Loop based control code verification and validation platform for the Delphi Solid Oxide Fuel Cell plant and control system has been developed. Verification activities are managed using the System Verification Manager tool. This paper outlines the application of this platform for safety and diagnostics verification and validation for a Solid Oxide Fuel Cell system.

INTRODUCTION

Control algorithms for complex industrial systems require thorough verification and validation against a variety of different operating conditions to ensure robustness and reliability of the design. Further, algorithm verification and validation becomes more important in the context of safety critical features of a system. Manual comprehensive testing of a development control algorithm can be a tedious, human-error-prone, time and resource intensive process, which is often infeasible.

This paper demonstrates an automated verification and validation process of the control system for Delphi Corporation's Solid Oxide Fuel Cell (SOFC) system. The performance requirements of the control algorithm are first identified and categorized in a hierarchical manner. Individual branches of the requirements hierarchy lead to verification activities. The verification activities are the actual executable tests that interact with the SOFC plant and the control algorithm for testing the requirements. These test executables are parameterized to enable reuse across multiple verification activities, which facilitates easier maintenance and robustness in the context of a development or evolving control strategy. The requirements testing is managed through System Verification Manager (SVM), a MATLAB based tool that allows hierarchical organization of requirements, management of verification activities, and individual and batch mode verification of the requirements [3].

The verification and validation of the safety and diagnostic features of the SOFC control algorithm in a Hardware-in-the-Loop (HIL) environment are demonstrated herein. In this environment, the real plant is emulated by a detailed mathematical model of the SOFC system. The HIL environment consists of the plant and the controller, running on separate real-time processors, and the wiring harness. The HIL environment not only gives a closer representation of a real system but also allows automated simulation and detection of electrical and hardware related faults. Use of similar HIL systems for studying different applications of fuel cells is becoming increasingly prevalent. Salem et. al [1] have demonstrated the use of a PEM fuel cell as a drive for a switched reluctance motor in a real-

time simulation environment. Dufour et. al [2] have simulated a PEM fuel cell based hybrid vehicle on a real-time platform. The increasing research in the area of fuel cells makes use of such HIL systems very cost effective and convenient.

This paper provides a background of SOFC systems and introduces the Delphi SOFC system. In the subsequent section we provide the motivation for control verification and validation and give a general outline of adopted testing procedures. Further we describe categorization of Verification and Validation tests and organization in System Verification Manager. This is followed by a detailed description of test logic development for a sample verification activity. HIL development for this activity is outlined in the following section, which includes discussion on model preparation, electrical interface development and communication interface development. Finally, concluding remarks are provided.

NOMENCLATURE

HIL : Hardware-In-the-Loop
S&D : Safety and Diagnostics
V&V : Verification and Validation
SDVV : Safety and Diagnostics Verification and Validation
SOFC : Solid Oxide Fuel Cell
SVM : System Verification Manager
ECU : Electronic Control Unit
HAL : Hardware Abstraction Layer
O/I: Operator Interface
CAN: Controller Area Network
I/O: Input/Output

BACKGROUND

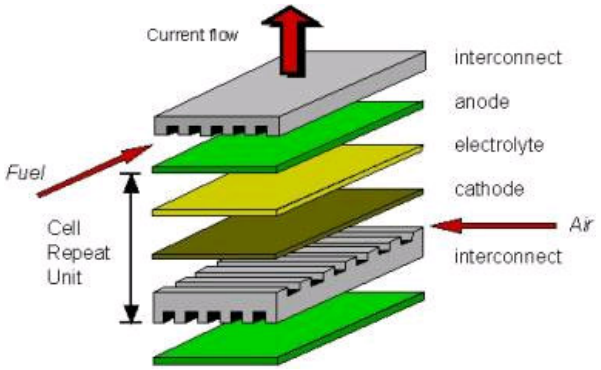

Figure 1: An elemental SOFC unit

An SOFC system is an energy conversion device that converts chemical energy to electrical energy at very high temperatures of 700-1000°C. The SOFC is considered to be the most desirable fuel cell for generating electricity from hydrocarbon fuels. The SOFC technology is simple, highly efficient, tolerant to impurities, and can partially internally reform hydrocarbon fuels. The fuel cell system is constructed by stacking elemental fuel cells. Each elemental fuel cell is composed of thin layers composed of an interconnect, an anode, an electrolyte, a cathode, and another interconnect, stacked in that order, as shown in Fig.1. The cell units are then layered into fuel cell stacks to match the system electrical architecture design.

In the membrane-electrode assembly (MEA) shown above, the electrolyte which is usually a polycrystalline ceramic, is an electrical insulator but an O^{2-} conductor. It is impervious to gas flow. The electrodes are normally composed of porous cermet composites. The interconnect functions as the electrical contact between the cathode and the anode and is the mechanism for collecting the current generated by each cell. The interconnect maintains high electrical conductivity and zero porosity. The interconnect is thus exposed to both the reducing environment of the anode and the oxidizing atmosphere of the cathode. The fundamental chemical equations that take place inside an SOFC are illustrated in Fig.2 [6-7].

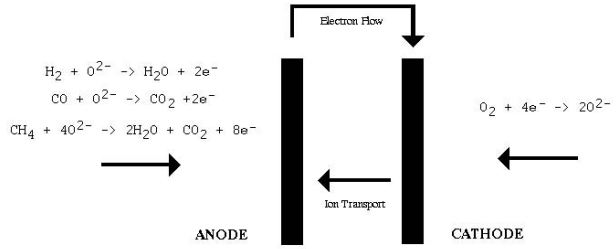

Figure 2: Stack electro-chemistry

The Delphi SOFC system consists of the fuel cell stack, the air delivery system, the fuel metering system, the power management system, and a waste energy recovery system [8-10]. The air and fuel delivery systems provide metered air and reformed fuel at necessary pressures and flow rates while assisting in maintaining an appropriate thermal environment for proper functioning of the fuel cell stack. The thermal conditions are critical since SOFC systems operate at high temperatures and have a significant thermal capacity.

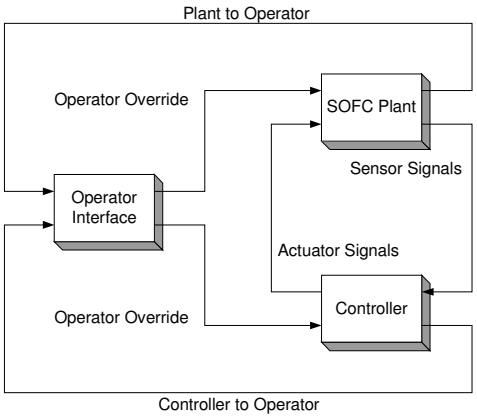

Figure 3: Delphi SOFC System and Control schematic

The Delphi SOFC system consists of a multitude of components and sub-assemblies whose functions are highly interdependent. Proper functioning of the system is achieved by a detailed control design which orchestrates state based calculations using unique algorithms that ensure desired thermal trajectories and flow levels. In addition, intricate state based sequences and calculations govern power management, system efficiency, and performance. The control design for this system is complicated due to heavy inter-modular and intra-modular interactions between subsystems in addition to the large number and variety of sensors and actuators in the system. This complexity results in a high volume of information flow to and from the controller as well as within the control algorithm itself.

It is evident from the discussion above, that an elaborate control algorithm is necessary for controlling the flow characteristics and pressure dynamics of the air and fuel flows, and thereby controlling the thermal characteristics and performance of the stack. The control algorithm for the Delphi SOFC system is modularized into different units, such as the Process Air unit, the Sensor Inputs units, Reformer Control and Waste Energy Recovery unit, the Stack Control unit, the Anode Tail Gas Recycle Control Unit, the Power Management unit, the Actuator Outputs unit, and the Executive Control unit. A detailed SOFC plant model is developed to create a completely virtual simulation environment. This is necessary for control code development, testing and tuning. The plant model represents the physical setup closely and interfaces with the controller using the same sensor and actuator signals. Additionally, there is an Operator Interface unit which interfaces with the control algorithm to provide sensor signal overrides, actuator signal overrides, and set-point/reference signal overrides. This feature is especially useful for control algorithm testing and tuning as well as system development and calibration. A schematic layout of these major modules of the Delphi SOFC system is shown in Fig.3.

VERIFICATION AND VALIDATION

Overview

Control code testing for industrial controllers is extremely tedious. The scope of testing is potentially very broad. This paper demonstrates an automated V&V approach for testing the S&D features of the SOFC controls. Safety considerations are imperative for a system that operates at temperatures of 800^{o}C and above, and is a source for large scale power demands in stationary and mobile platforms. Hence very detailed S&D logic is designed and embedded within the system controls. The S&D algorithm is designed to detect faults in plant modules, faults in sensor data, faults within external devices, and controller faults. The S&D logics are dispersed in the individual modules of the control algorithm to detect faults with a high granularity. A hierarchical structure of the S&D further ensures thorough data capture from individual modules and supervisory actions based on various fault triggers.

A generic approach has been adopted for the SDVV effort which is applicable for all verification. The fundamental steps of this common approach are enumerated below. Within the framework of this approach, different categories of tests function differently to address their respective requirements.

1. Allow the fuel cell system to reach an appropriate or meaningful state where a particular verification activity is relevant.
2. By polling several signals from the system and control algorithm and by reading relevant control parameters, determine the value of the sensor signal that would cause the fault that is being tested. It is reasonable to assume here that the particular sensor signal that will be directly responsible in triggering a specific fault is known from the knowledge of the control design itself.
3. Apply this sensor injection to the plant model.
4. Monitor the control algorithm and verify that upon detecting the fault, the control algorithm reacts to this fault in accordance with the S&D design, i.e. takes appropriate steps, flags appropriate signals, follows a desired logic sequence, to maintain safety of the plant.
5. Revert to original system settings, i.e. revert to default control parameters if changed, reset the plant model, and prepare for the next V&V activity.

Categorization and Organization in SVM

The V&V tests are categorized and hierarchically organized using the SVM (System Verification Manager) tool. The S&D requirements testing that have been demonstrated for the DELPHI SOFC system are categorized as follows:

1. Sensor range tests
    a. Upper limits,
    b. Lower limits,
2. Control loop tests
    a. Loop saturation tests,
    b. Loop lock tests,
    c. Integrator saturation tests,
3. Hardware failure tests
    a. Component X fault test,
    b. Component Y fault test, etc.

Category 1 consists of tests that verify and validate the portion of the S&D code that responds to sensor readings that are abnormally high or low. This is important for two purposes. First, an abnormally high or low sensor reading may indicate departure from normal operating conditions of the fuel cell system, or a serious failure, or impending damage to the plant. Secondly, it could be a signal for abnormal data acquisition or a problem in the wiring harness. In either case the S&D control logic is programmed to take precautionary measures in such situations. The V&V activities in category 2 are in connection with feedback control loops. A number of feedback control loops function simultaneously in the SOFC control algorithm. The generic structure of control loops is given in Fig.4. The

control loops function by comparing a sensor reading to a reference signal or set-point value and applying a Proportional-Integral action on the error to generate an actuation command. Three different fault modes are specified for each control loop. They are:

a) Loop saturation fault
b) Loop lock fault
c) Integrator saturation fault

Loop saturation fault is triggered when a control loop effort is saturated. Loop lock fault is triggered when an error signal does not return within specified limits within a specified time interval, after the error has crossed a limit. The integrator saturation fault is triggered when the integral term of the Proportional Integral control is saturated. Evidently, all the three fault modes mentioned above are indeed scenarios where a feedback control loop will not be functioning, at least partially. Portions of the S&D logic monitor each control loop to detect the faults mentioned above. Finally, the tests in category 3 are related to verify and validate the S&D logic that are designed to detect specific hardware failures and respond accordingly.

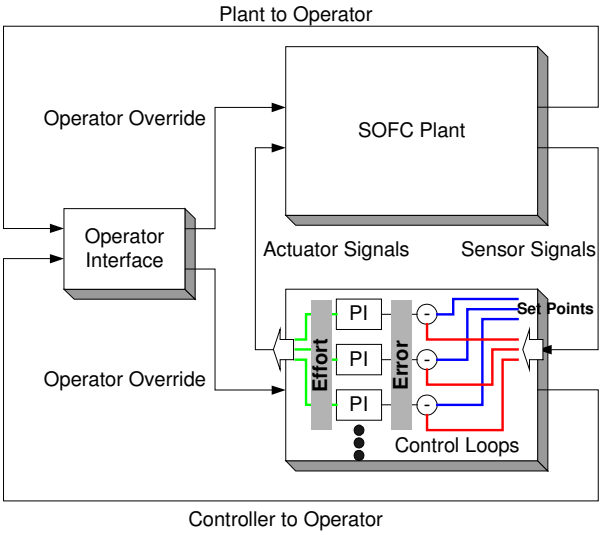

Figure 4: Control Loops schematic

For each category, a generic verification method is designed and implemented. A Verification method is an executable program that actually performs a specific test following the generic five step approach listed in the previous section. Thus, considering the categorization above, we have one verification method for each category 1.a, 1.b, 2.a, 2.b, and 2.c. For example for category 1.a. all sensor range tests for upper limit are executed through one verification method. A certain amount of customization is unavoidable and it is primarily due to the very nature of the control design and uniqueness of plant model components. Category 3 for instance is unique and requires a specific verification method for each test since different hardware components have different characteristics and functionalities.

The broad S&D requirements mentioned above are managed using SVM, which is a MATLAB based tool. SVM is a tool for hierarchical specification of requirements. Some of its features are verification activity management (enable/disable requirements, batch mode verification), verification result management (through verification status messages, and test data post processing facilities), and an extensible and open framework that admit flexible definition and reuse of models and verification methods. The requirement hierarchy is first entered in SVM. All branches of the requirement hierarchy end with a verification activity. The verification activity is essentially a function call to a verification method. In Fig.5 we show the hierarchical organization of different SDVV requirements in SVM. The highlighted item in the list represents a verification activity. Once the verification methods are tested individually, each category can be exhaustively tested by SVM. This feature of SVM reduces testing time as compared to the manual effort that it takes to do the same task. There is an initial effort involved in developing the verification methods but the advantage lies in their reusability with minimal modifications as the control design itself evolves.

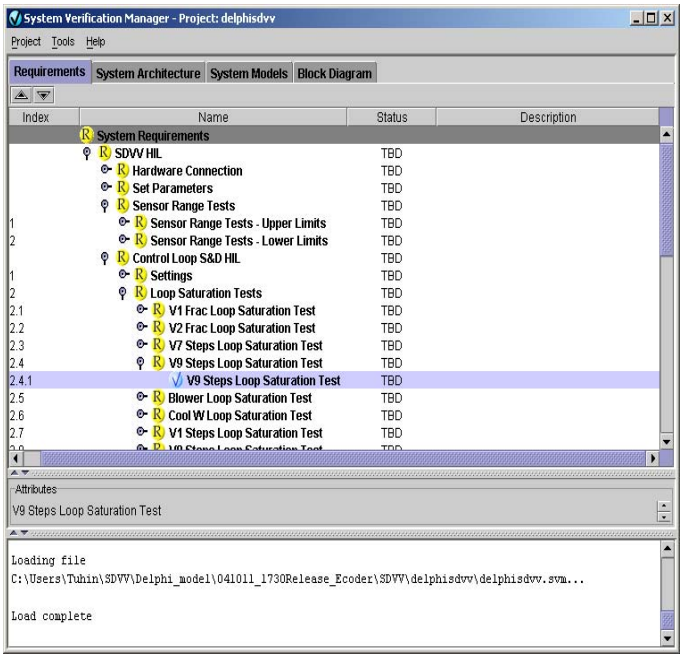

Figure 5: SDVV setup in SVM

For use with SVM, the verification methods are written in MATLAB m-code format. Verification methods are initially registered into SVM. During the registration process, all the arguments of the verification method are identified by SVM. When a verification method is associated with a verification activity, these arguments appear in parametric form to the SVM user. This forms the basis of parameterization of the verification

activities. If the same verification method is associated with multiple verification activities, as is the case for similar SDVV tests, then the same parameter set will be presented to the SVM user and by entering different values of the parameters, the user is able to run a series of similar verification activities. Furthermore, during verification method registration, the data types of all the arguments are registered. This allows SVM to identify and flag incorrect use of a verification method. A parameter list for a loop saturation SDVV test for one of the SOFC control loops is shown in Fig. 6.

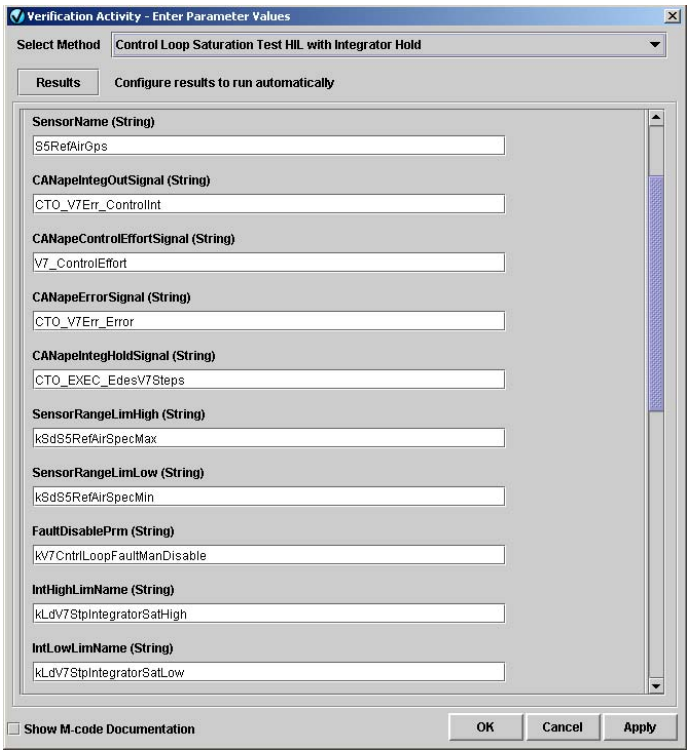

Figure 6: Verification method parameterization

Note that the visible parameters in Fig.6 are string arguments. This is because several actual parameter names and signal names that are relevant for this test are entered here. The verification activity monitors these signals and performs get and set parameter operations on the ECU in course of execution of this verification activity. This idea is applied for all the tests done for the SDVV effort.

Verification Method Development

As mentioned before, a significant advantage of our verification activities development is the parameterization of the verification methods. Additionally, the verification methods are programmed to intelligently determine proper sensor injections necessary to trigger a fault. In this section we outline the algorithm for determining sensor injection for loop saturation tests. The corresponding analysis for other test categories is omitted. The following nomenclatures and conventions are used in the derivations of the sensor injection choice for the loop saturation test:

r : Set point (target) of the PI loop
x : Measured sensor value
x_{inj}: Injected sensor signal
x_l : Minimum sensor signal value that can be detected by the ECU
x_h : Maximum sensor signal value that can be detected by the ECU
P : Parity of the control loop

$$P = \begin{cases} 1 & if \quad error = (desired - measured) \\ -1 & if \quad error = -(desired - measured) \end{cases}$$

e : Loop error $(= P\,(r - x))$
K_p : Proportional gain of the control loop
K_i : Integral gain of the control loop
L_h : Loop higher saturation limit
L_l : Loop lower saturation limit
I_h : Integrator higher saturation limit
I_l : Integrator lower saturation limit
e_{acc} : Lower limit of the maximum acceptable error
L_{ff} : Loop feed-forward estimate
f_h : Integrator hold level fraction
f_r : Integrator reset level fraction
F : Loop effort
F_I : Loop integral effort
F_P: Loop proportional effort

The loop integral effort F_I and loop proportional efforts F_P are given by

$$F_I = \begin{cases} I_h & if \quad K_i \int e\,dt > I_h \\ & K_i \int e\,dt \\ I_l & if \quad K_i \int e\,dt < I_l \end{cases} \tag{1}$$

and

$$F_P = K_p\, e \tag{2}$$

The fundamental control loop equation is then:

$$F = \begin{cases} L_h & if \quad (L_{ff} + F_p + F_I) > L_h \\ & (L_{ff} + F_p + F_I) \\ L_l & if \quad (L_{ff} + F_p + F_I) < L_l \end{cases} \tag{3}$$

The loop saturation test can be done in four different methods:

1. Test the upper loop saturation limit using the proportional gain and repeated integrator resets to prevent integrator saturation.
2. Test the upper loop saturation limit with the integrator held at a level so that the integral effort is within

saturation limits and the proportional effort is sufficient to cause loop saturation.

3. Test the lower loop saturation limit using the proportional gain and repeated integrator resets to prevent integrator saturation.
4. Test the lower loop saturation limit with the integrator held at a level so that the integral effort is within saturation limits and the proportional effort is sufficient to cause loop saturation.

With method 1, the desired loop error should satisfy

$$e_{h1} \geq \frac{L_h - L_{ff}}{K_p}, \quad and \quad e_{h1} > 0 \tag{4}$$

With method 2, the desired loop error should satisfy

$$e_{h2} \geq \frac{L_h - L_{ff} - I_h + (1-f_h)|I_h|}{K_p}, \quad and \quad e_{h2} > 0 \tag{5}$$

With method 3, the desired loop error should satisfy

$$e_{l1} \leq \frac{L_l - L_{ff}}{K_p}, \quad and \quad e_{l1} < 0 \tag{6}$$

With method 4, the desired loop error should satisfy

$$e_{l2} \leq \frac{L_l - L_{ff} - I_l - (1-f_h)|I_l|}{K_p}, \quad and \quad e_{l2} < 0 \tag{7}$$

With methods 1 and 3, the integrator reset is done whenever the integral effort satisfies:

$$I_h - F_I < (1-f_r)|I_h| \quad or \quad F_I - I_l < (1-f_r)|I_l| \tag{8}$$

With methods 2 and 4, integrator is held whenever the integral effort satisfies

$$I_h - F_I \geq (1-f_h)|I_h| \quad or \quad F_I - I_l \geq (1-f_h)|I_l| \tag{9}$$

The desired errors derived above are also limited by the maximum and minimum errors possible due to the current set point. The maximum and minimum errors are dependent on the parity of the loop and are given by:

$$e_{\max} = \begin{cases} r - x_l & for \quad P = 1 \\ x_h - r & for \quad P = -1 \end{cases} \tag{10}$$

$$e_{\min} = \begin{cases} r - x_h & for \quad P = 1 \\ x_l - r & for \quad P = -1 \end{cases} \tag{11}$$

Assuming that $e_{\max} > 0$ and $e_{\min} < 0$ we use the following procedure to determine the sensor injection:

$$x_{inj} = r - P\frac{\max(0, e_{h1}) + e_{\max}}{2} \quad if \quad e_{h1} < e_{\max} \tag{12}$$

$$x_{inj} = r - P\frac{\max(0, e_{h2}) + e_{\max}}{2} \quad if \quad e_{h2} < e_{\max} \tag{13}$$

$$x_{inj} = r - P\frac{\min(0, e_{l1}) + e_{\min}}{2} \quad if \quad e_{l1} > e_{\min} \tag{14}$$

$$x_{inj} = r - P\frac{\min(0, e_{l2}) + e_{\min}}{2} \quad if \quad e_{l2} > e_{\min} \tag{15}$$

In the implementation, the inequality conditions in (12) through (15) are checked and the sensor injection corresponding to the first satisfied inequality is applied for loop saturation test. Prior to the sensor injection, there is another preparatory step where normalized variations in the set point of the loop are checked. If wide set point variations are detected, the set point is overridden to a fixed value and then the test is performed. Also note that during this test the loop lock fault is prevented by setting a high minimum saturation on the acceptable error, i.e. by setting e_{acc} to a very large number.

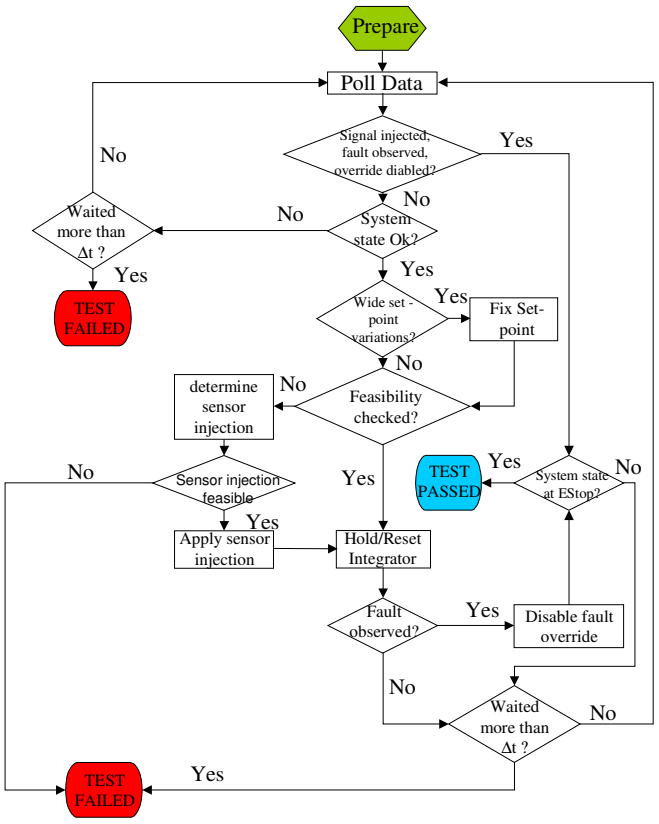

Figure 7: Typical test flowchart

The flowchart for the loop saturation test is shown in Fig.7. The essential steps in the execution of a verification activity are captured here. It can be seen that a typical test is initiated by a setup and feasibility test part. This is followed by a logic execution to determine the desired sensor injection. Next the sensor injection is applied and subsequently a series of tests are performed to validate the relevant S&D logic. Throughout the

execution of the steps mentioned above, the verification method frequently polls data and parameter values.

HIL SETUP

Overview

The control logic, which consists of the system control as well as the S&D logic, should be thoroughly tested for safe operation of the SOFC system under a variety of operating conditions. A simulation model of the SOFC system is used for preliminary checks and for verifying and validating the basic capabilities. However, this level of testing is insufficient and not thorough enough. While desktop runs help in the initial tuning of control parameters, a prototype system is ideal for more detailed testing and validation. However, considerations such as cost of fabricating a prototype, lead time, risks of component failure, etc, are factors that hinder quick repeatable testing and experimentation. An HIL platform is a very flexible and cost effective alternative to the two extremes. For the Delphi SOFC S&D V&V system, the HIL setup consists of the control logic running on a Delphi production intent controller, the SOFC ECU, and the plant model running on a real-time QNX operating platform from OPAL-RT [5].

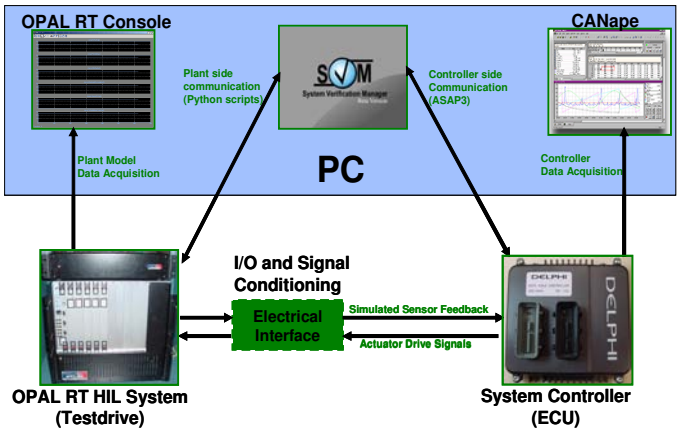

Figure 8: Schematic of the DELPHI SOFC HIL system

The real-time simulation platform is shown schematically in Fig.8. The host PC where SVM runs, communicates with the ECU through a CAN bus interface and with the QNX real-time target through TCP/IP. A wiring harness provides the sensor and actuator interface between the ECU executing the control algorithm and the QNX machine running the plant model.

A significant advantage of the HIL system is the ease with which the real plant can be emulated using a plant model. The fidelity of the plant model is an important consideration. The Delphi SOFC plant model is a high fidelity model that has been developed through extensive efforts alongside the control development process. The plant model performs similarly to the actual system. The model has evolved through extensive correlation and calibration efforts. Moreover, the plant model is used for desktop control algorithm development and coarse calibration of the control strategy. The plant model is ideal for HIL applications and serves as an ideal candidate for the SDVV efforts.

Plant Model Preparation

The control algorithm for the Delphi SOFC system is extensively tested and calibrated with the actual SOFC system. Hence the controller module and control algorithms were ready for use in a real-time simulation environment. The plant model, however, had to be prepared to complement the existing electrical interface of the controller ECU. The preparation tasks involved adding the low level hardware I/O software (HAL) to imitate the actual plant interfacing with the controller. Also appropriate modifications were necessary to add capabilities of sensor injection which is an integral part of the SDVV effort.

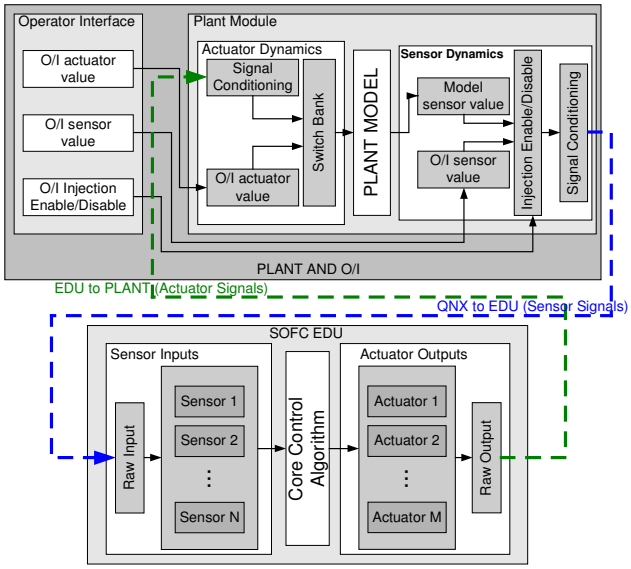

Figure 9: Model setup for SDVV tests on HIL

A block diagram showing the interfaces between the plant model and the controller are shown in Fig.9. The dashed lines indicate the actual electrical connections which represent the wiring harness in the HIL setup. The plant and Operator Interface modules are shown in Fig.9. The core plant model remains unchanged for the HIL setup. Modifications are made to the 'Actuator Dynamics' and 'Sensor Dynamics' blocks. In 'Actuator Dynamics', the HAL and additional logic are added to the 'Signal Conditioning' subsystem. An operator override feature for these signals is added in the form of the 'O/I actuator value' subsystem. A switch is implemented to facilitate switching between these signals.

Of primary interest is the 'Sensor Dynamics' block where a feature to facilitate sensor injection is implemented. The subsystem 'Model sensor value' carries the sensor signals generated by the plant model. Sensor injection is facilitated through the subsystems 'O/I sensor value' and 'Injection

Enable/Disable'. The former receives sensor signal overrides from the O/I. The later contains a switch bank to select or deselect this override value. The final sensor value is sent through the 'Signal Conditioning' module to the controller. The sensor injection process outlined above is automatically done during the execution of a verification activity through the communication interface between SVM and the plant model running on the target QNX machine.

Also note that the HAL interface for the plant model was developed using RT-LAB libraries provided by OPAL-RT. The library contains a comprehensive set of library elements that can be used for analog I/O, digital I/O, decoding PWM signals, resolving quadrature pulses, etc. The electrical and HAL interface setup is discussed in some detail in the following section.

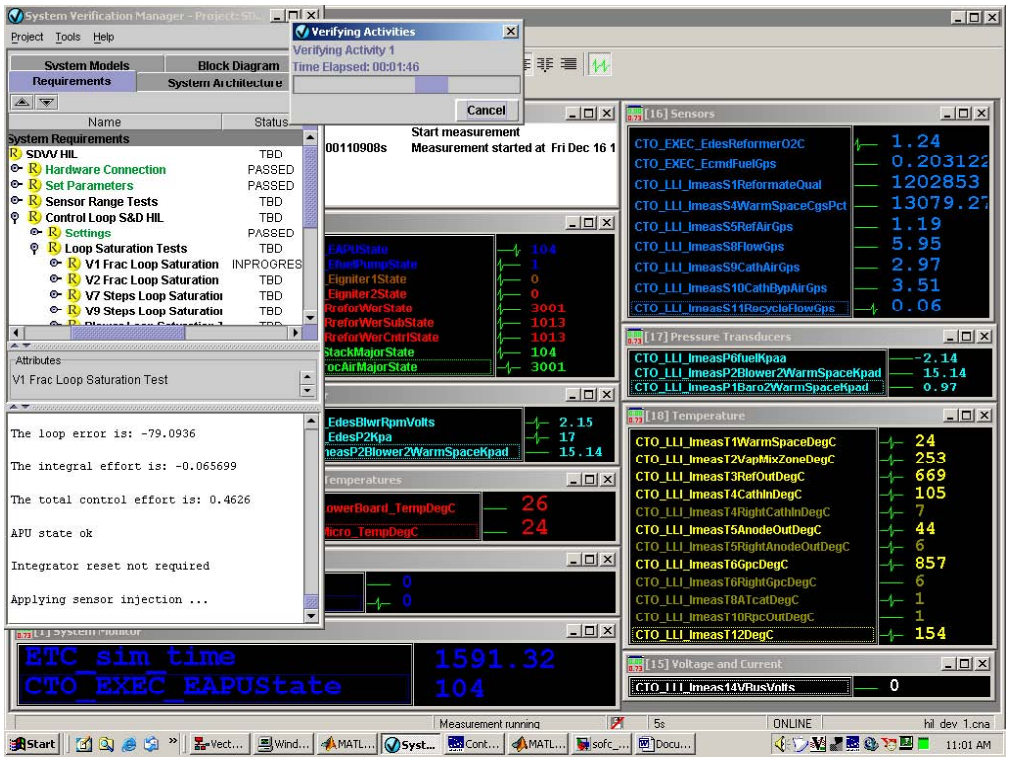

Figure 10: A sample SDVV test in progress

Electrical Interface and HAL

In setting up the HAL interface for the plant model, several preliminary tests, calibrations, and other setup tasks were done before the electrical interface was ready for real-time simulation and testing. The analog sensor signals from the OPAL-RT system to the ECU were calibrated. These channels carried the temperature, flow, and pressure sensor data from the plant model to the ECU. A calibration model was created for this purpose. Each channel was calibrated to match the commanded voltage and the measured voltages. The calibration map was linear for all the channels.

The ECU issues desired step commands to the stepper motors that are used to control the valve openings. These commands are sent by the ECU as quadrature pulses. Decoding of these quadrature pulses was necessary in order to use the plant model in a real-time environment. The decoding of quadrature pulses is critical since inaccuracies in decoding can easily accumulate and result in significant errors in flows and an eventual loss of control. Decoding of the quadrature pulses is done in two steps. The first step is to use the 'Quadrature Resolver' library element, which is provided in RT-LAB as part of the RT-Events library. The output of this block is rotation in degrees. The conversion of degrees (rotational coordinate) to steps (linear coordinate) is done through separate logic. This logic takes the saturation of steps between zero (closed valve) and a maximum (fully open valve) into account.

The fuel metering commands for flow are PWM signals issued by the ECU. The PWM signals are decoded using the 'RTE Period Meter' library element within the RT-Events library. Apart from the above mentioned tasks, there were additional verifications and confirmatory tests that were needed for the digital I/O.

Communication Interface

Finally, we discuss the communication interface between SVM and the real-time processors. In executing a verification activity, SVM communicates with the ECU and the QNX real time target

for actions such as signal monitoring, get and set controller parameters, perform sensor injection, etc. For visualization we use CANAPE which communicates with the ECU via CAN to provide online display of signals available within the controller. Both SVM and CANAPE run on a host PC that communicates with the ECU and the OPAL-RT system. The communication between SVM and ECU is established via the same CAN line that is used by CANAPE. The communication between SVM and the ECU is established using the 'ASAP3 Translator', a utility within CANAPE which converts ASAP3 codes into CANapeAPI calls. The CANapeAPI calls in turn have direct access to the EDU. The ASAP3 codes are generated from MATLAB using the ASAP3 toolbox which is developed by Emmeskay Inc [4]. SVM and the OPAL-RT system communicate using RT-LAB's Python APIs which can be called using MATLAB functions that are available within SVM.

In Fig.10, a screenshot of a sample test run is shown. In the background, the CANAPE screen is visible. The CANAPE screen shows streaming measurement data from the ECU. The SVM screen is on the foreground at the top left hand corner. In this example, a loop saturation test is in progress. The message window of SVM can be used for displaying messages such as the current status of the test using text messages that are programmed within the verification method. The time elapsed in a test is displayed on the progress window of SVM which appears to the right of the main SVM window.

CONCLUSIONS

In applications of fuel cell systems, a high level of performance can be delivered though a carefully designed and a thoroughly tested control algorithm. The requirements to be tested are usually numerous and hence manual testing is tedious and error prone. In this paper we demonstrate an automated validation and verification platform for the safety and diagnostics algorithm of an SOFC control system. The requirements are managed within SVM, which executes verification activities in a real-time environment. The basis of each verification activity is sensor injection that causes a desired fault to be triggered. The verification method then monitors the control algorithm to determine if the requirement is satisfied. This automated testing platform results in an approximate 10x reduction in testing time, compared to manual testing. An important feature of this test platform is the parameterization of verification methods that admits reuse of verification methods across similar verification activities.

ACKNOWLEDGMENTS

The authors thank Gopalan Raghavachari of Emmeskay, and member of the SVM development team, for his help with the use of SVM during the SDVV test platform development. In addition, the authors thank Kevin Keegan and Stephanie Cornelius of Delphi Corporation for their hard work and contributions in making this project successful.

REFERENCES

[1] M. Salem, T. Das, X. Chen, S. Akella and S. Sivashankar, "Real Time Simulation for Switched Reluctance Motor Powered by A Fuel Cell System", Proc. ASME-Power Conference, PWR2005-50090, 2005.
[2] C. Dufour, T. Das, S. Akella, "Real Time Simulation of Proton Exchange Membrane Fuel Cell Hybrid Vehicle", Global Power-train Congress, Ann Arbor, MI, September 2005.
[3] "System Verification Manager", *as seen on website,* http://www.ece.cmu.edu/~webk/svm/.
[4] "Controller Communication Toolbox", *as seen on website,* http://www.emmeskay.com/index.php?option=com_content&task=view&id=83&Itemid=146
[5] http://www.opal-rt.com/
[6] S. Mukerjee, K. Haltiner, S. Shaffer et al "Development of a Solid Oxide Fuel Cell stack by Delphi and Battelle" , SOFC IX –J. Electrochem Soc., Quebec City, May 2005.
[7] S. Mukerjee, S. Shaffer et al., "Development of a SOFC Stack by Delphi and Battelle", in Solid Oxide Fuel Cell VIII, S.C Singhal and H. Yokokawa, editors, PV-2003-07, p-88, The Electrochemical Society Proceedings Series, Pennington NJ (2003).
[8] S. Mukerjee et al.: "Solid Oxide Fuel Cell Auxiliary Power Unit – A new paradigm in Electric Supply for Transportation", in Solid Oxide Fuel Cell VII, S.C Singhal and H. Yokokawa, editors, PV-2001-16, p-173, The Electrochemical Society Proceedings Series, Pennington NJ (2001).
[9] J. Zizelman, S. Shaffer, S. Mukerjee, "Solid Oxide Fuel Cell Auxiliary Power Unit- A Development Update"; SAE World Congress, Paper 2002-01-0411, Detroit, March 2002
[10] J. Zizelman, C. DeMinco, S. Mukerjee, J. Tachtler, J. Kammerer, P. Lamp, "Auxiliary Power Units with SOFC Technology for Independent Electric Power Supply in Passenger Cars", European Solid Oxide Fuel Cell Conference Proceedings, 2002, Lucerne, Switzerland.