White Paper

Using Portable Stimulus for the Closed-Loop Specification and Verification of ISO 26262 Functional Safety Requirements

Version: 1.2
October 2018

Breker Confidential

Abstract

Translating requirements and safety goals required by the ISO 26262 automotive safety standard into practical detail for development purposes is still a somewhat manual process. The Accellera Portable Test and Stimulus Standard (PSS) shows great promise for automating this flow. It can provide a machine-readable mechanism for describing requirements and safety goals, generating rigorous tests, and producing reliable coverage metrics at an effective level of abstraction. These test may be leveraged throughout the verification flow for both systematic verification and random fault analysis. This paper builds on previous work to show an example of how PSS may be applied in this manner, leveraging a closed-loop implementation-to-verification methodology.

Breker Verification Systems, Inc.
www.brekersystems.com

Copyright Breker Verification Systems, Inc. ©2018 All Rights Reserved

Introduction

The ISO 26262 Automotive Safety Standard specifies how safety goals are to be evaluated in an automotive device development process. There is some vagueness in the standard on exactly how these goals are to be implemented and verified, and how the effectiveness of this process is measured. The Accellera Portable Test and Stimulus Standard (PSS) offers new techniques particularly applicable to this problem. The use of PSS can revolutionize how the implementation of requirements in general is verified and how the effectiveness of this process is measured. It may also improve the ability to create and reuse tests at different levels of abstraction during the random fault analysis flow.

The ISO 26262 Systematic Development Flow

The ISO 26262 Automotive Safety Standard [1] specifies two central verification processes for hardware design. First, the Systematic Development Process is concerned with ensuring that the requirements for the hardware design are specified correctly, have been fully considered with safety in mind, and that the implementation of these requirements has been fully and rigorously verified. Second, the Random process is concerned with ensuring that the correct operation of the device is maintained even if internal components are affected due to environmental or other effects.

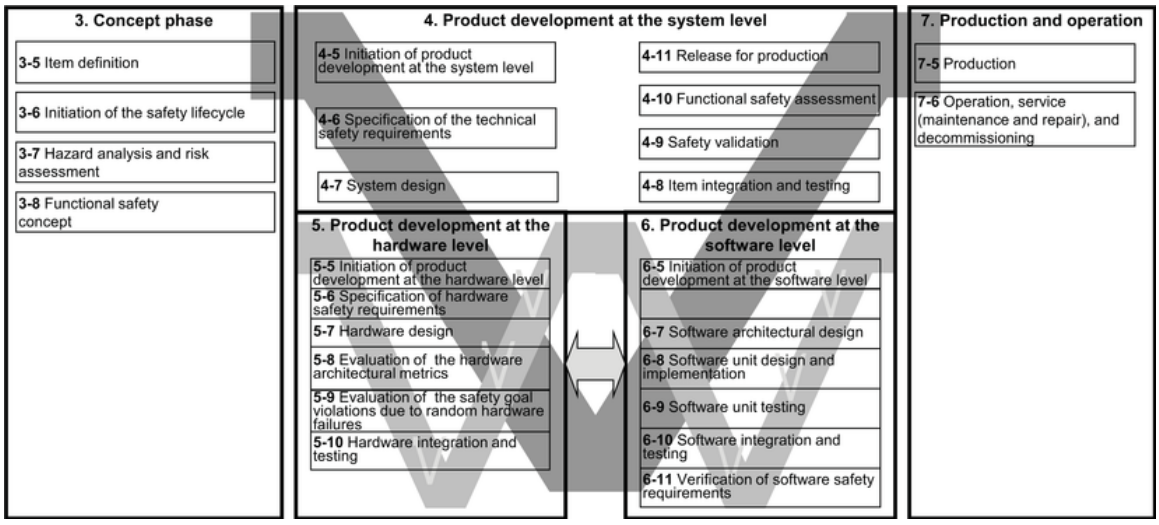

Figure 1: ISO 26262 Systematic Development Process describes a rigorous mechanism for detailing requirements that must be met. Courtesy: ISO

The Systematic Development Process describes a rigorous mechanism for detailing requirements, starting with system-level concept definitions, through safety requirements specifications, system design, and finally hardware and software product development (Figure 1). Depicted as the well-known "V-Model," the right-hand side of the V deals with equivalent verification processes that match the left-

hand specification sections. The verification processes feed back information from right to left to demonstrate a complete testing of the respective left-hand elements.

The Automotive Safety Integrity Level (ASIL) is a grade applied to a device that indicates the level of its compliance to the ISO 26262 safety requirements. Four levels are detailed: A, B, C and D, where D is the safest. A device that would have a significant impact on the safety of the users — as measured by the potential severity of injuries, exposure to injury, and device controllability — must have an ASIL D classification. To achieve this, a rigorous requirements management process, together with the exhaustive verification of those requirements, must be demonstrated.

Verifying Systematic Requirements

Automotive component requirements that must meet the ISO 26262 ASIL D standard are usually detailed in a requirements management tool or at least a comprehensive spreadsheet. The requirements are decomposed into individual specification items or features (Figure 2).

Each one of those features has associated with them a verification goal, and these are then translated into a verification plan, which describes the required tests, coverage metrics, and verification mechanism. It should be noted that each requirement has its own set of verification goals and metrics that must be measured on an individual basis. Interleaved with the functional requirements are safety goals that must be met as part of the development process.

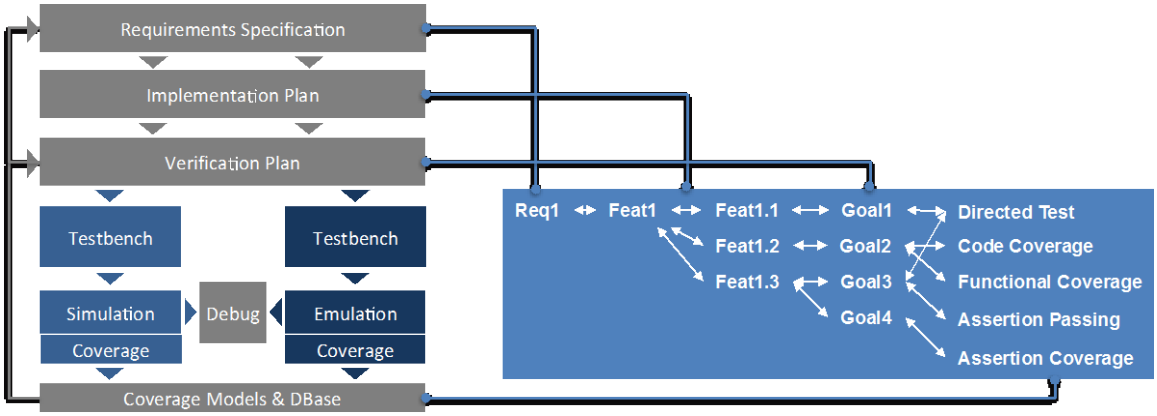

Figure 2: Decomposition of requirements into development processes as individual specification items or features. Source: TV&S

Safety mechanisms designed to correct errors introduced during the operation of the device (due mainly to environmental effects) are included in the design itself. The Systematic Verification Process must ensure that these safety mechanisms do not interfere with the correct operation of the device.

In addition, a significant analysis of the device is performed to ensure that it will self-correct in the event of a Random error. This involves injecting faults at points in the design during a special verification process, and then observing the continued operation of the device. Once tests have been derived for the Systematic phase, they may also be leveraged for this Random analysis process. Being able to leverage the same tests from the block level right up to full system can save a lot of time during this phase.

In [2], Beckers et al. described a structured method whereby ISO 26262 safety goals may be broken down into functional safety requirements, and how these may be represented using a unified modeling language (UML) notation. The ISO 26262 standard provides little guidance for how these goals might be effectively broken down into requirements. It does, however, note that a rationale for the requirement must be provided together with evidence that any assumptions made were handled appropriately, any attributes related to the requirement type were defined, that the requirement is implementable, and that the implementation was fully tested.

The notion of using UML to describe these requirements in a machine-readable manner is aligned with the idea of creating an executable specification that may be leveraged by a verification environment, where coverage metrics can be evaluated directly against the specification. The Accellera Portable Test and Stimulus Standard provides a method to describe such a specification, automatically produce tests and coverage metrics from it, and directly compare the specification with the verification outcome. This has the potential to produce a closed-loop, automated process to evaluate ISO 26262 requirements.

An Overview of Portable Stimulus

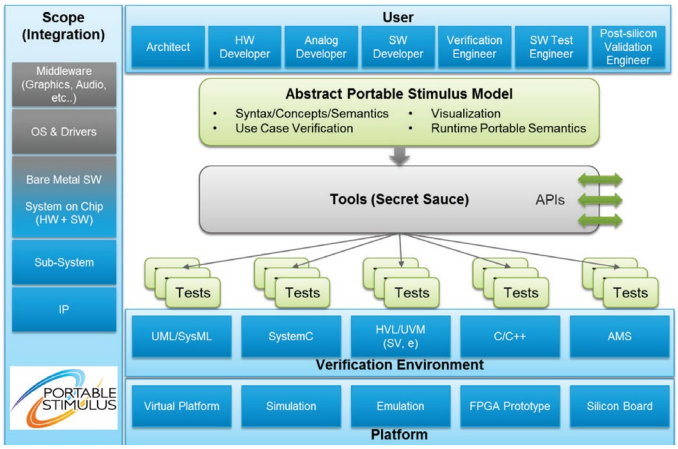

Figure 3: The Portable Test and Stimulus Standard operates across the verification flow. Courtesy: Accellera Systems Initiative

In June 2018, the Accellera Systems Initiative, the Electronic Design Automation (EDA) industry standards body, produced the Portable Test and Stimulus Standard, version 1.0 [3]. This standard provides two semantically equivalent formats to specify verification intent in such a way that it may be leveraged in multiple tools and methodologies throughout the semiconductor verification process (Figure 3).

Models are written at a level of abstraction designed to produce easily comprehendible test processes that may be applied at the system level.

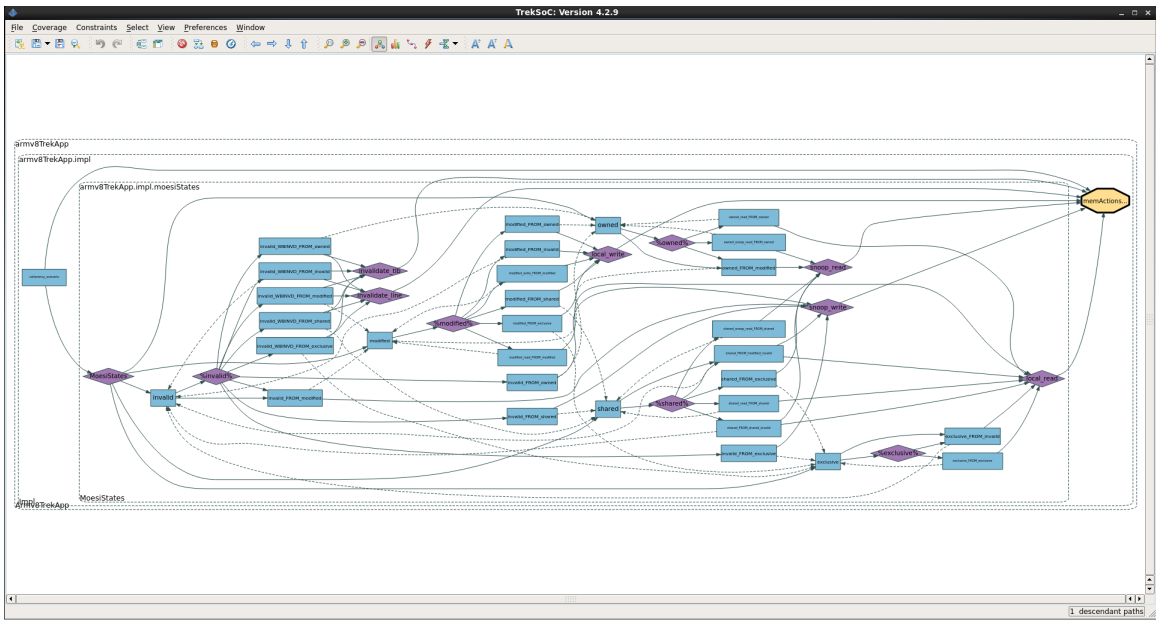

Figure 4: Cache Coherency MOESI States can be displayed as Portable Stimulus Graph.

Portable Stimulus is a significant development in the EDA industry, and a number of EDA vendors and end users are making use of this standard. It has already been proven to provide a clear depiction of specifications from which specific verification scenarios may be extracted by an associated tool. Furthermore, some of the tools that use the standard can generate comprehensive test sets and coverage models based on the specification that may be correlated to the modeled scenarios. This enables a closed-loop test system that closely mirrors the ISO 26262 recommended procedure.

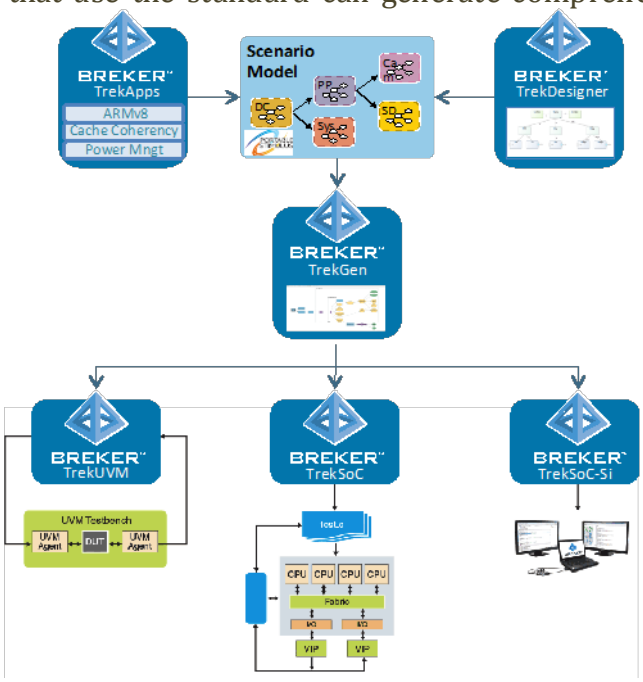

Figure 5: Breker's Trek tool suite provides deployment flows across all verification phases.

Portable Stimulus models may be displayed as graphs in a format that closely resembles UML (Figure 4). A Portable Stimulus tool reads this description and produces a test set by walking the graph and solving randomized actions. This process involves test synthesis, system-level constrained random solving, test scheduling, and other process elements. The test sets may then

be deployed into different testbenches depending on the verification phase for which they were intended.

Although various tools exist today, Breker has produced a complete verification solution along these lines with deployment flows across all verification phases (Figure 5).

Given the work described in [2], a machine-readable, automated flow can be provided that makes use of Portable Stimulus to replace a number of error-prone manual tasks in a typical safety-critical, automotive semiconductor development program. This paper describes such a flow and uses the same example of a steering column lock as in [2].

Using Portable Stimulus to Specify Safety Requirements

In [4], Beckers et al. continue their work on providing a model to perform a risk assessment of automotive safety requirements using UML. They provide a framework with which to analyze safety goals and strategies to achieve those goals that align with the ISO 26262 standard.

They use an example of a steering lock controller that we will continue to explore in this paper. Figure 6 shows the safety goal structure for the Electronic Steering Column Lock (ESCL).

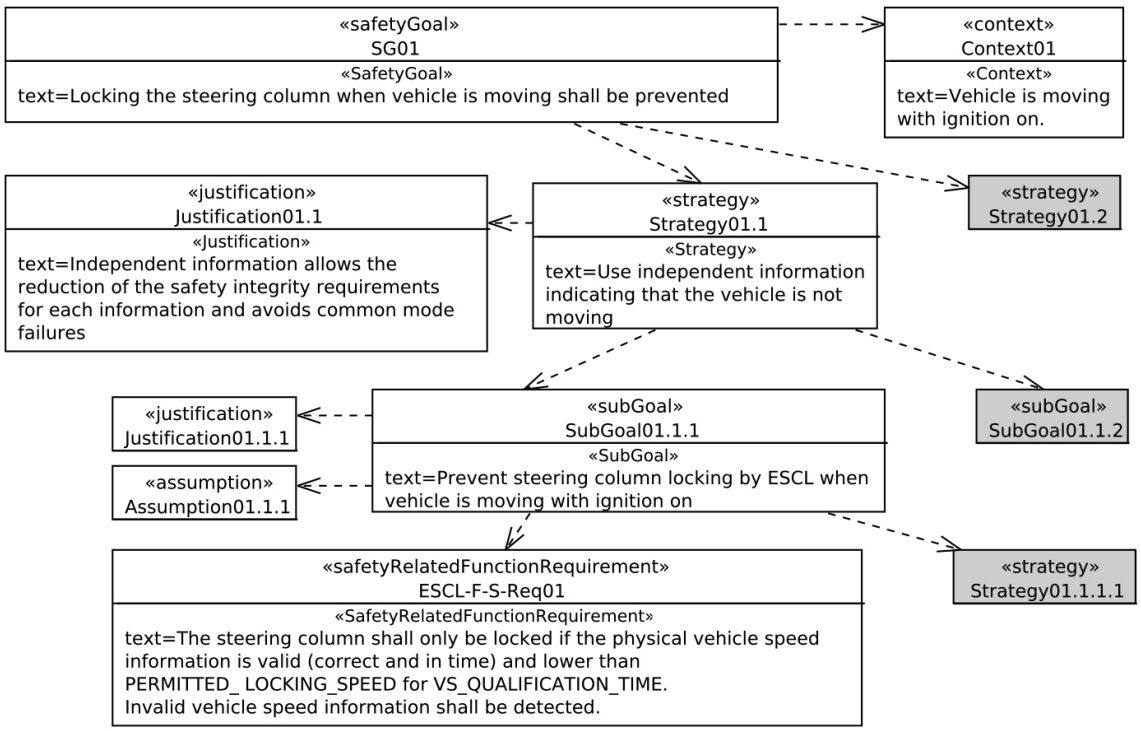

Figure 6: The Safety Goal Structure is a framework to analyze the safety characteristics of a Steering Column Lock [2].

A simple specification for this hypothetical ESCL unit might include the following:

- The steering column lock is set to lock when the ignition key is not present (note the key could itself be electronic and activate the ignition remotely).
- If the key is present and the ignition is switched on, the ESCL should be unlocked.
- To prevent the ESCL locking when the car is moving, an independent test of movement should be used to prevent common mode failure.
- The specification could go on to consider the car moving if the key is present and the ignition is not switched on, in the case of an engine malfunction for example. This may be accomplished in a manual locking system.
- Also to be specified is the moment at when the ESCL should lock, when the ignition is switched off, or when the key is not detected.

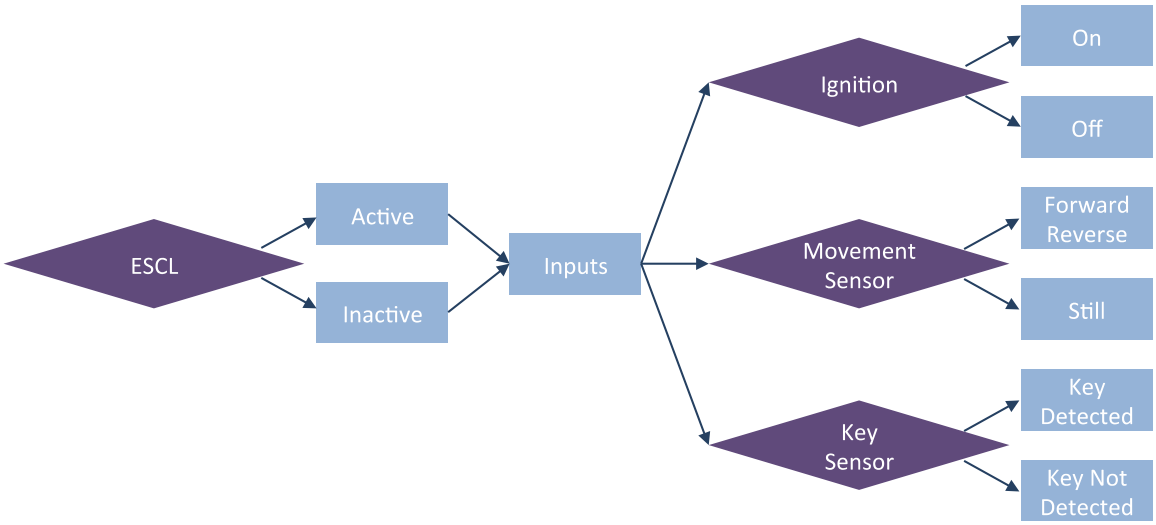

Figure 7: The ESCL specification items described using a PS graph, including select goals in purple and sequence goals in blue.

Figure 7 shows how the items of such a specification could be described using a Portable Stimulus graph. Note the purple diamonds are "select goals" and the blue rectangles are "sequence goals."

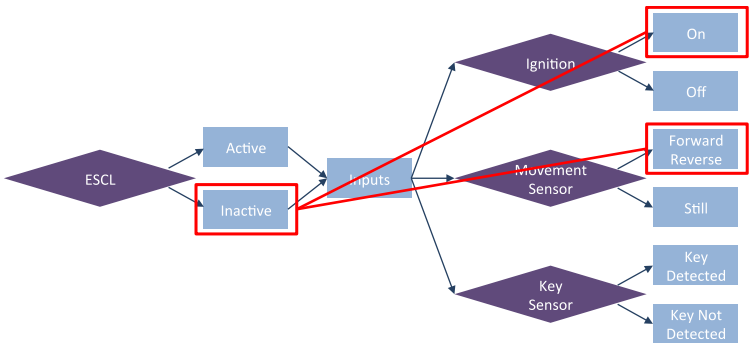

Figure 8: The ESCL Graph with Path Constraints defines the safety goals.

The graph is constructed by first selecting the possible outcomes and then choosing inputs that could lead to these outcomes. Because the graph includes both the result and inputs to the test, it is possible to generate self-checking

tests and the coverage model for the entire scenario. Different inputs may be randomly selected to attempt to find a corner case that produces the wrong result.

Having created this simple model, it is now possible to define the safety goals, and this can be done using a modeling feature known as a "Path Constraint." Path Constraints allow relationships between the graph nodes to be specified, and these are then included in the generated tests. For example, the safety subgoal "prevent steering column locking by the ESCL while the vehicle is moving and the ignition is on" may be exactly translated into a Path Constraint as shown in Figure 8. A pseudo description of the constraint might be:

(ESCL := Inactive) IF (Movement Sensor == Forward-Reverse) && (Ignition == On)

It should be noted that, while Path Constraints were discussed by the Accellera Portable Stimulus Working Group, they were not included in the first version of the standard. Breker has included this feature in its product line, and it may be applied to standard models.

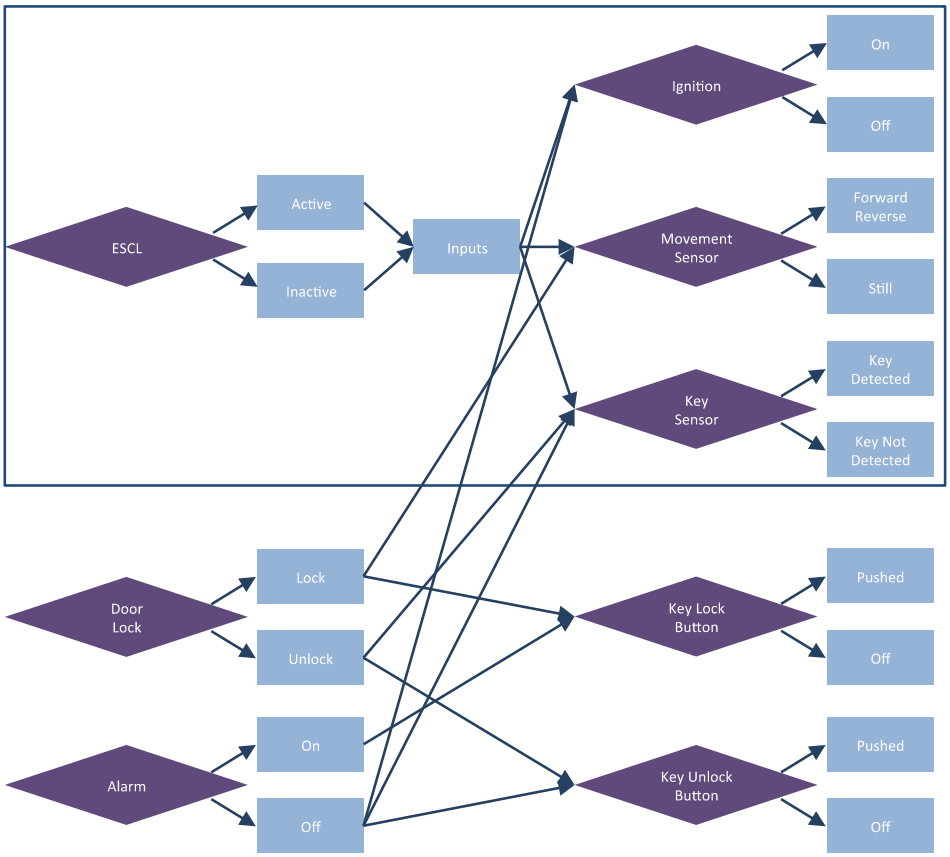

Figure 9: Extending the graph to add additional specification items can be done with Portable Stimulus.

Although this is a fairly simple example on its own, it is easy to connect scenario segments to create a much broader specification. Indeed, one of the advantages of Portable Stimulus is the modular nature in which full specifications may be built up from individual scenarios. For example, imagine extending the specification to include unlocking the doors and shutting off the alarm. Figure 9 demonstrates the beginning of this extended graph.

As shown, the modular nature of the graph allows for the system-level specification to extended, with additional constraints used to specify additional safety subgoals. For example, additional constraints could include the door locking automatically if the car is moving and unlocking in the event of an airbag deployment.

Furthermore, as the microcontroller subsystem is developed, more detailed tests may be easily overlaid into the graph to provide safety checks of the subsystem as well as wire harness integrity checks.

Reusing Portable Stimulus Tests for Random Fault Analysis

One of the major advantages of Portable Stimulus is the easy reuse of tests at different levels of design abstraction and throughout the verification process. In addition, manipulating test controllability is simplified, given the more white-box approach of Portable Stimulus over other constrained random methods such as the Universal Verification Methodology (UVM).

Random fault analysis consists of running tools such as fault simulation, sometimes combined with formal verification, to observe the effect on device operation of inserting a fault at specific locations in the design. Fault simulation requires input stimulus, and the better this stimulus is at targeting faults, the more efficient the analysis will be. As fault simulation is a compute-intensive process, efficient testing is key.

As noted, to achieve an ASIL D rating, >99% fault coverage must be achieved right across the final design. To achieve these results in practice, a design team will perform various analyses at different verification levels, starting at the block level and processing through subsystem and full system test. For example, a team might perform a verification step at the block level just to make sure a single fault will correctly trigger an alarm signal, waiting for the subsystem level to run rigorous fault analysis to fully exercise the safety mechanisms, and then perhaps perform a Monte Carlo statistical analysis at the full system level. Being able to reuse the same stimulus, as well as checkers and coverage models, at all three of these levels saves time in both crafting the tests to best target the fault analysis and comparing the coverage at the different abstraction levels.

Automating Portable Stimulus Comprehensive Test Generation

One of the most valuable characteristics of the Portable Stimulus test methodology is the enablement of white-box testing using abstract random variables and paths.

Traditional constrained random testing with simulation through standards such as UVM targets inputs of the Device Under Test (DUT) and provides sequences of tests in an attempt to activate the entire device to drive a high-coverage metric. This is difficult to do and can result in many wasted test sequences in an attempt to get the DUT into a state where meaningful verification can take place.

Formal Verification has the opposite characteristic in that the entire state space of a DUT can be explored using expected properties written as assertions. These assertions can be used to describe entire specification elements and safety goals, but the technology has limitations that allow it to handle only a limited number of state transitions on a large design.

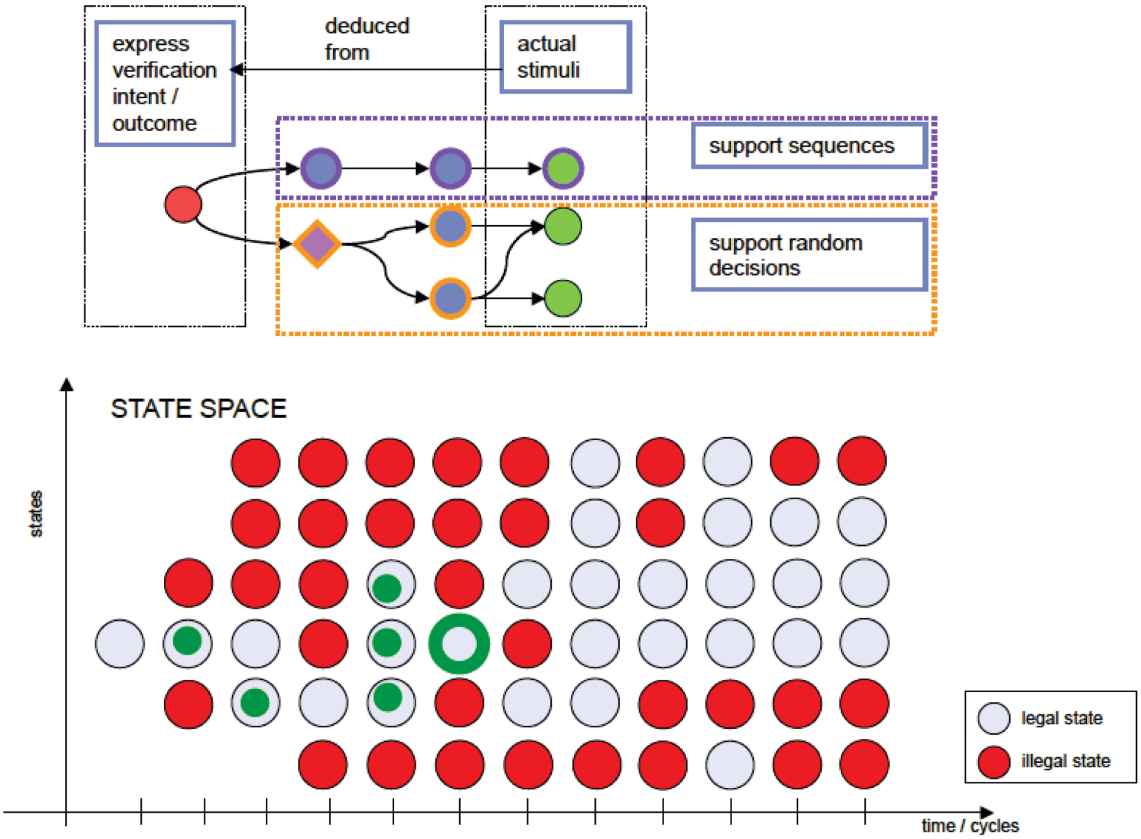

Figure 10: Graph-based test modeling enables deep state-space exploration [5]. Source: IBM

Portable Stimulus enables a middle ground between these two approaches. It allows test sequences to be specified to activate the full behavior of the DUT as efficiently as possible, but makes use of dynamic methods such as simulation and emulation to execute large designs and test sets (Figure 10).

In [5], a team at IBM explores the use of Portable Stimulus to target deep state spaces in a methodical and rigorous fashion. From its paper:

"…graphs are a great vehicle to describe the legal set of states through a combination of directed sequences (where state set is sparse) and random decisions (where the state space opens up)."

It is this principle that makes this method so effective for generating rigorous, high-coverage test sets for safety-critical applications.

Essentially the graph approach allows for easy definition of abstract directed tests that can walk through a sparse state space, combined with randomized selections in areas where a broader state space must be explored. This white-box style of depicting directed or randomized actions mirrors the testbench requirements of safety-critical devices.

Once the graph is composed and constraints set, the tool will walk the graph applying the constraint solver as required to generate sequences of tests. These test sequences are run through a scheduling synthesis tool that takes into account the resources required to interleave series of test sets together to be applied across the design inputs. Outcomes contained in the graph are also used to provide checks.

Closed Loop Verification and Coverage Assurance

One of the most powerful aspects of the Portable Stimulus approach is the effective measure of scenario coverage metrics.

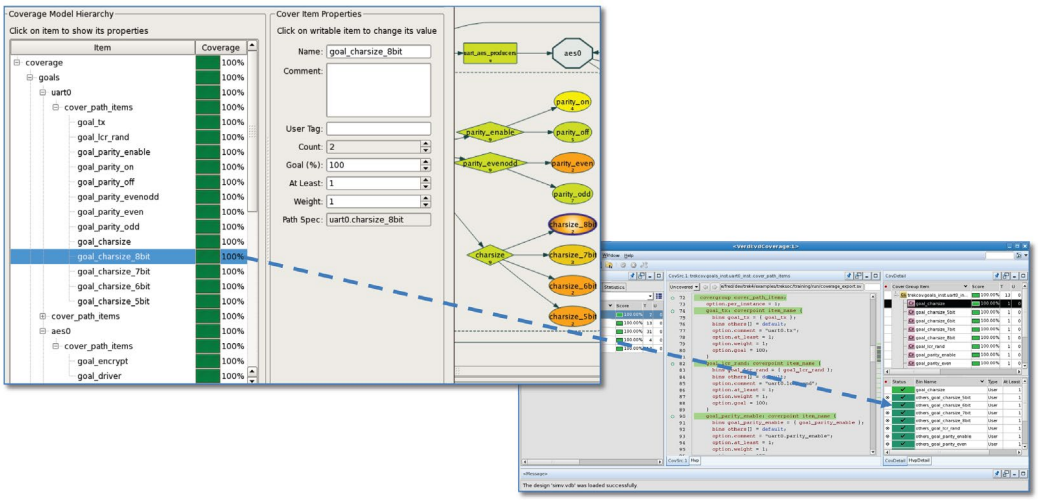

Figure 11: Breker's coverage display has a direct connection to a requirements management tool for reporting purposes.

Having constructed a graph, a Portable Stimulus tool should be able to provide a clear measure of nodes of the graph that have been explored, and areas of the graph that are still untouched. The Breker tools are also able to automate coverage closure by allowing the end user to select an unmet coverage metric and direct the tool to provide a test that targets this specific metric (Figure 11). A good Portable Stimulus tool, such as Breker's Trek suite, will also be able to identify paths and nodes on the graph that are unreachable and use test execution time to profile a design for critical bottlenecks.

An important aspect of measuring coverage on a graph is the ability to select paths across the graph that must be covered per the safety goals and cross coverage points where different parts of the graph need to be checked. Similarly to Path Constraints, Path Coverage is not yet included in the standard, but is part of the Breker tool suite.

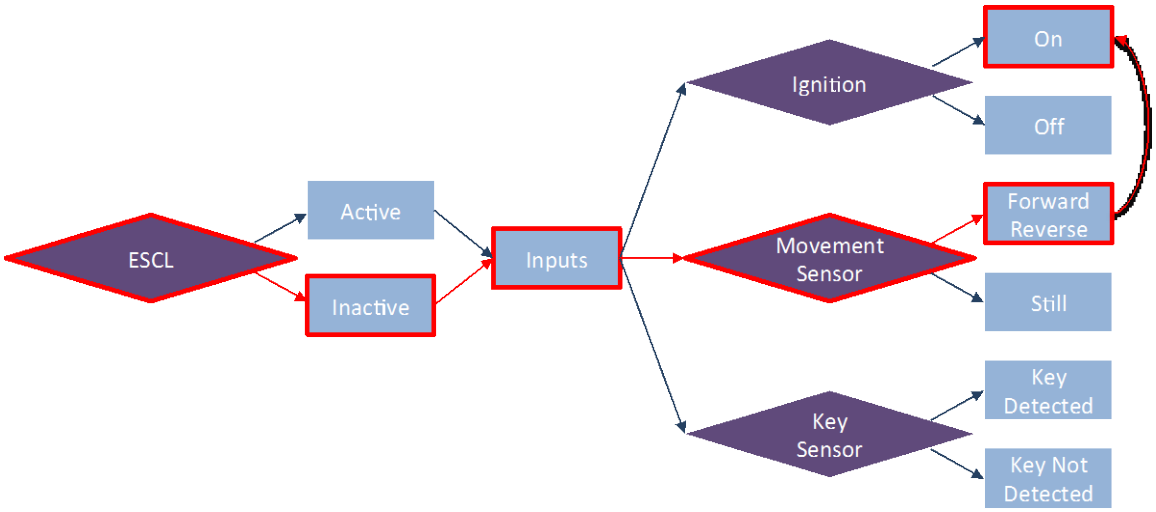

Figure 12: Path Coverage is used to ensure the safety goals are met.

It is possible to set up a Path Coverage goal that requires a test to be executed, ensuring the ESCL is inactive when the car is moving cross checked with the ignition switch set to on (Figure 12). Similarly, coverage goals can be set to check items that should not occur, such as ensuring the column lock is never set to active in these same conditions. Once these safety goals are specified, a measure will be provided showing when these are covered and by which tests. This information can be output in a variety of formats for reporting purposes, including a direct connection to a Requirements Management tool, as shown in Figure 11.

Summary

This paper reviewed an ISO 26262 framework for describing requirements and associated safety goals, illustrating that these goals can be directly translated into Path Constraints used with a Portable Stimulus tool such as Breker's Trek suite. It highlighted that this method of modeling scenarios lends itself to the rigorous

verification required of automotive devices, and that PSS correctly applied can automate a large part of the verification flow and required coverage feedback.

References

[1] The International Organization for Standardization (ISO) 26262 Standard, Parts 4,5,8 https://www.iso.org/obp/ui/#iso:std:43464:en

[2] Beckers K., Cote I, Frese T, Hatebur D, and Heisel M. Systematic Derivation of Functional Safety Requirements for Automotive Systems. Proceedings of the 33rd International Conference on Computer Safety, Reliability, and Security - Volume 8666.

[3] The Portable Test and Stimulus Standard, version 1.0. The Accellera Systems Initiative.
www.accellera.org/images/downloads/standards/pss/Portable_Test_Stimulus_Standard_v1.0.pdf

[4] Beckers K., Cote I, Frese T, Hatebur D, and Heisel M. A Structured and Model-Based Hazard Analysis and Risk Assessment Method for Automotive Systems. 2013 IEEE 24th International Symposium on Software Reliability Engineering (ISSRE).

[5] Korrapati S., Horbach H., Keuerleber K., Jung A., IBM. Walking the Graph, A Holistic Approach to Graph Based Verification for Logic With Sparse State Space. www.brekersystems.com/resources/case-studies