SYSTEM LEVEL FAULT INJECTION SIMULATION USING SIMULINK

Wai Yuen Tang[1] Marcelo Mizuki[2] Mitch Norcross[3] Fengying Qiao[4]

[1]15 Trafalgar Square, Nashua, NH, wta@melexis.com. [2]15 Trafalgar Square, Nashua, NH, miz@melexis.com. [3]15 Trafalgar Square, Nashua, NH, mno@melexis.com. [4] Transportstraat 1, 3980 Tessenderlo, Belgium, qfe@melexis.com

Abstract - **Fault injection is a popular methodology for verifying the design of fail-safe systems. It gives insight into how the system behaves in presence of a fault. There are different ways to implement fault injection. Depending on the nature of the product and project phase, one approach is more favorable than others. In the ASIC design phase, the simulation method is preferred because it is otherwise difficult to insert faults onto a Silicon die. In this paper, we present a functional model of a LIDAR system for fault injection simulation. The model is created using the Mathworks Simulink tool based on the design specification. The safety mechanisms of the system are also included in the model. The model can run in either mission mode or diagnostic mode. Various analog faults were injected into selected blocks of the model, to evaluate the diagnostic coverage of the HW architecture metrics and also the completeness/correctness of the implemented safety mechanism. In the end, the fault injection campaign at model-level successfully identified the weaknesses of the system architecture, and a new safety mechanism has been added to improve the diagnostic metrics. Moreover, compared to fault injection at analog circuit level, the Simulink model can be reworked quickly and easily for a new design. This methodology can be applied to other functional safety semiconductor systems with similar topologies.**

Keywords- Fault injection, Model-based design, Simulink, ASIC design

I. INTRODUCTION

Over the last decades, the semiconductor industry has provided a steady improvement in the safety of automobiles. Advances in modern electronics have accelerated the number and features of safety systems. In automotive electronic systems, functional safety and reliability are critical requirements for the design. International Organization for Standardization ISO-26262 "Road vehicles-Functional Safety" provides appropriate standardized requirements, and processes an automotive-specific risk-based approach to determine integrity levels, also known as Automotive Safety Integrity Levels or ASILs [1]. ASILs are used to specify applicable requirements of the ISO-26262 standard so as to avoid unreasonable residual risk. There are four ASILs defined by the ISO-26262, from the lowest integrity level to the highest, being ASIL A, ASIL B, ASIL C, and ASIL D.

In case there is fault in the system that would lead to the violation of the safety goal, e.g. corrupted data, safety mechanisms should be implemented to detect the fault and report to the host device within a specified fault detection and handling time interval. Some examples of safety mechanisms are parity bit checking on memory unit and voltage monitors checking the level of power supply. Safety mechanisms can be specific to a subset of the system. They can also be applied to a full system, where a known test input is used and the output of the system is examined. For automotive electronics devices that are used in safety-critical applications, it is important to have a fail-safe design such that the device always indicates to the host when faults are present.

ISO-26262 strongly recommends fault injection methods as a part of the hardware-software integration and testing for integrity of ASIL B or higher [1]. Fault injection is a test aimed at studying the system behavior under fault condition. Fault injection can be implemented in three ways: Hardware-implemented, Software-Implemented, and Model-Implemented [2]. Hardware-implemented fault injection uses physical perturbation such as pins forcing, and radiation to analyze the circuit behavior with certain fault condition. Software-implemented fault injection uses software to alter the timing and/or data of the hardware. Both of these methods require the design in the form of physical device (actual product or prototype). For the ASIC design, the physical system is generally not available during the design phase. Therefore, ASIC designers heavily rely on simulation during the design phase. In simulation, the design is represented by a model and can interact with external stimulus. The design under test (DUT) and the test environment are simulated on computer systems. As described in the ISO-26262, depending on the purpose, fault injection can be implemented at various levels of abstraction (e.g. semiconductor component top-level, part or sub-part level, RTL, etc [1]. Selection of the abstraction level can depend on the nature of the fault, the required accuracy, etc. If the model correctly reflects the DUT, then with lower-abstraction levels models, we can evaluate the DUT better (with higher accuracy). However, the drawback is that it consumes more engineering time of developing the model and computation time for running the simulation.

Currently, there are CAD tool software solutions that perform fault injection simulations in an automated and exhaustive way. For example, Z01X by SYNOPSYS [3] is one of the solutions in the market. It is designed to analyze the DUT performance under fault condition and categorize the results based on their seriousness and visibility. This tool is used in a Melexis ASIL C project in 2015. A significant reduction of implementation and simulation time is reported as compared to manually perturbing the DUT circuit netlist. However, Z01X can only simulate digital faults. An additional method is required for faults in analog and mixed-signal domains.

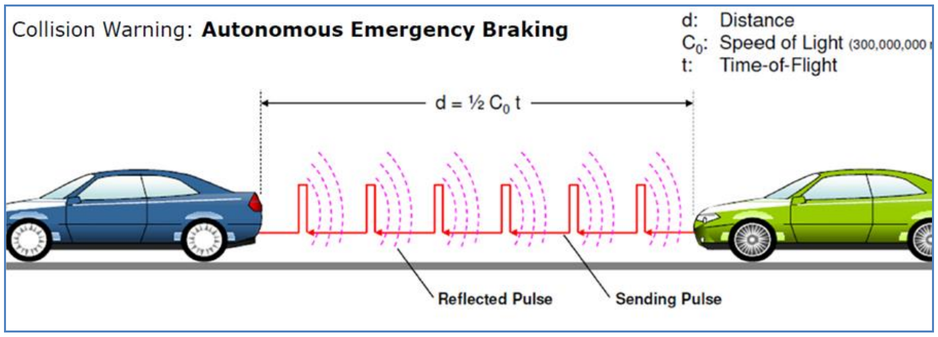
For analog simulations, faults can be injected at transistor level. Some common fault examples are a digital gate output stuck at a certain values permanently and a net that is open or shorted. The challenge of transistor level fault injection is that it can be a time-consuming process for large-scale design. Moreover, fault injection at either RTL level or transistor level can only be performed at the later stage of the design phase, when the circuit or RTL code is available. Therefore, in this paper, we will focus on the Model-Implemented Fault Injection, which can be performed at the earlier stage of the design. Besides, based on the output of the ISO26262-compliant failure analysis performed on our system, which will be explained in later section of this paper, fault injection at top-level is sufficient to evaluate the metrics.

In order to study the effects of faults on the DUT without considering each and every transistor in the design, an abstract model is needed. In this paper, the Mathworks Simulink is used for building a functional model of the design. Simulink is a model-based design tool that allows engineers to focus on the behavior of the design. Model-based design is a common methodology for complex design. It provides a common design environment during the feasibility and early design phase. Both the normal function operation and the safety mechanism can be evaluated using model-based design. It can be created and modified efficiently compared to the transistor level model.

Other research has proven the feasibility of fault injection on Simulink model for an embedded control system in the automotive industry [4]. The results in paper [4] suggest that fault injection on the models is efficient and useful to assess the design's robustness. Consistent results are found when comparing model-implemented fault injection to other fault injection methods.

II. IMPLMENTATION

In our LIDAR project, which requires the integrity level of ASIL-B, the Mathworks Simulink environment is used to model the design. The mission for the design is to measure the distance of objects by emitting and receiving light \ and measuring the round trip delay of light reflected by the objects in the scene. LIDAR has numerous advantages compared to its counterparts. It has a higher resolution than radar; it provides extra distance information as compared to image sensor. Applications of LIDAR include autonomous emergency braking, blind spot monitoring and self-driving cars.


Collision Warning: **Autonomous Emergency Braking**
d: Distance
C_0: Speed of Light (300,000,000
t: Time-of-Flight
$d = \frac{1}{2} C_0 t$
Reflected Pulse
Sending Pulse

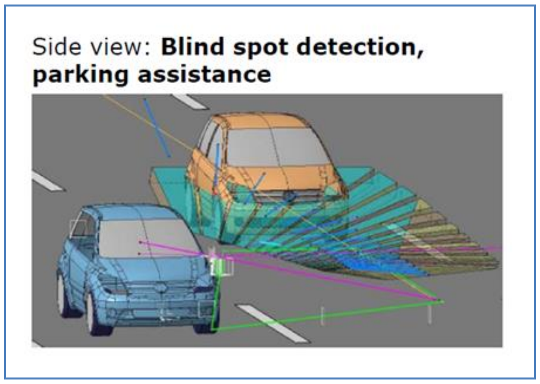


Figure 1. Examples of LIDAR applications in automotive.

In this project, the system can operate in mission mode, where the target object distance is measured; It can also operate in diagnostic mode, where a known test input is fed into the system. The system is designed to work with a host (master) device, which configures the system and receives distance measurement and error status from the system.

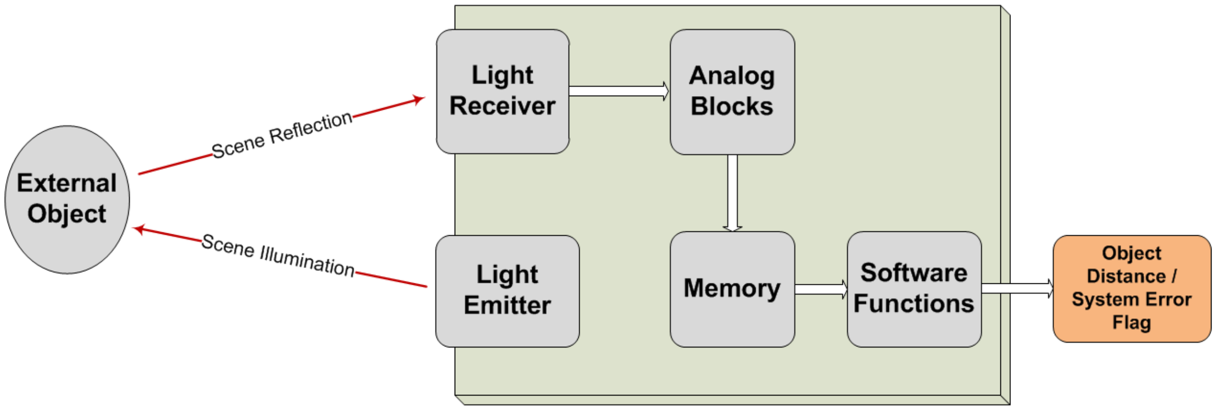


Figure 2. Simulink model diagram for LIDAR (in mission mode).

The Simulink model is initially intended for system level modeling for the mission mode operation on this project. The model contains two parts: the target object and the device under test (DUT). In our case, the DUT is the LIDAR system. The LIDAR system can be further divided into sub-blocks based on the actual design. There are some analog blocks to amplify and digitalize the data. Also there are some firmware functions to compute object distance from the digitized data. The model diagram is illustrated on Figure 2 above.

In order to evaluate the HW metrics and residual failure rates of the system with the highest risk, a failure modes, effects, and diagnostic analysis (FMEDA) has been performed. FMEDA is required by the ISO-26262. For ASIL-B applications, ISO recommends the single point fault metric (SPFM) to be >=90% [1]. FMEDA is a quantitative down-top safety analysis, and the level of detailed analysis depends on the level of implementation of the safety mechanism. The workflow of the FMEDA analysis is defined by the ISO26262, and can be simplified as shown in figure 3. In FEMDA, each parts/subparts was analyzed with the possible random HW faults. For example, a sub-parts such as ADC could have possible failure of accuracy error, offset error, linearity error. As shown in the FMEDA workflow, for each specific failure mode of the specific subpart, the failure effects at the system level are further analyzed. If this failure mode could lead to the violation of the safety goal or safety requirement at the system level, the related safety mechanism such as voltage monitor or built-in-self-test to detect the specific failure mode is investigated. The diagnostic coverage will be estimated based on the efficiency of the safety mechanism.

The final SPFM and Residual Failure rate will be determined by adding up the failure rate and diagnostic coverage of each individual safety related parts/subparts.

After the above steps, we can already estimate the SPFM of the system and also identify the possible weak parts of the system (the parts with higher failure rate and/or lower diagnostic coverage).

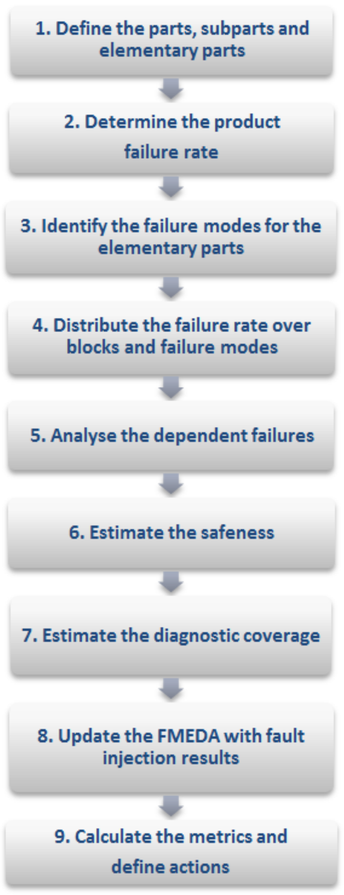

1. Define the parts, subparts and elementary parts
2. Determine the product failure rate
3. Identify the failure modes for the elementary parts
4. Distribute the failure rate over blocks and failure modes
5. Analyse the dependent failures
6. Estimate the safeness
7. Estimate the diagnostic coverage
8. Update the FMEDA with fault injection results
9. Calculate the metrics and define actions

Figure 3. FMEDA workflow.

Once the weak parts/subparts of the system are identified by the FMEDA, the list of fault injection actions can be defined. As mentioned before, the abstraction level of the fault injections depends on the purpose of the analysis. For the LIDAR system in this paper, potential faults in the listed weak subparts are covered by safety mechanisms at the system level. In this case, the fault injection evaluation at the system top level is sufficient for the safety analysis.

With the defined actions for fault injections on the LIDAR system, we can further refine the Simulink model to support fault injection and detection. The corresponding block models are modified with higher accuracy based on the block level design specifications. Signal noise is also introduced to each block. The faults are realized using a multiplexer to switch to a modified (faulty) signal at key points in the targeted block. Individual faults can be enabled or disabled by the model user. A generic fault injection implementation is shown in Figure 4:

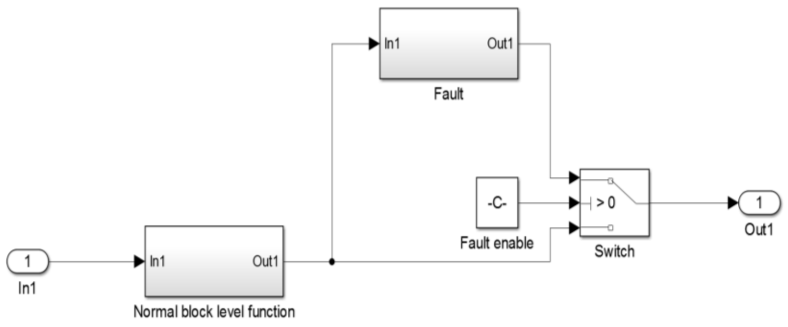
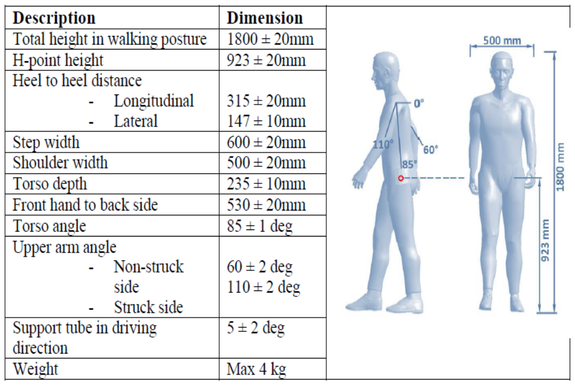


Figure 4. Fault injection block in the Simulink model.

The next step is to add diagnostic mechanisms to the model. In the actual diagnostic mechanism, some known test patterns are fed into the receiver. The diagnostic software examines the characteristics of the data after analog, digital, and firmware processing against some pre-defined expectations. If the data examined during the diagnostic mode do not meet the expectation, an error status is set and reported, and the system transitions into a pre-defined safe state, which is detectable by the host. This notifies the host that the hardware may be unreliable.

To simulate the system as it will operate in the target application, some object models are created. Pedestrian and car models are chosen. The models are created based on the European New Car Assessment Programme (Euro NCAP) test standard [5]. The document introduces a standard test protocol for autonomous emergency braking system. The models used on NCAP test are well-defined on the target section. Important parameters are the reflectivity and the overall dimensions. The models used in our simulations are further simplified as rectangles.



Description	Dimension
Total height in walking posture	1800 ± 20mm
H-point height	923 ± 20mm
Heel to heel distance - Longitudinal - Lateral	 315 ± 20mm 147 ± 10mm
Step width	600 ± 20mm
Shoulder width	500 ± 20mm
Torso depth	235 ± 10mm
Front hand to back side	530 ± 20mm
Torso angle	85 ± 1 deg
Upper arm angle - Non-struck side - Struck side	 60 ± 2 deg 110 ± 2 deg
Support tube in driving direction	5 ± 2 deg
Weight	Max 4 kg

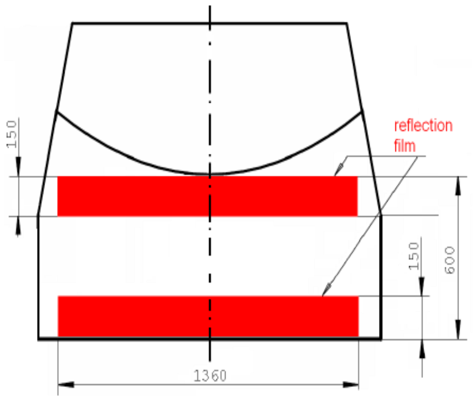


Figure 5. Target model specifications from Euro NCAP test standard [5].

The Simulink experiment is setup in a static way, meaning the distance between the target and the system is unchanged during the experiment. The targets are placed within the system's detectable range. An individual fault is injected to the system before simulation is started, and the outputs of the DUT (distance and system error status) are observed. If a fault causes a false object detection or prevents a real object detection during mission mode, we define it as a dangerous fault. The magnitude of the fault which starts causing the mission mode failure is recorded. If the diagnostic mode does not detect and report the fault with equal or smaller fault magnitude, we define the fault as undetected. Otherwise, it is a detected fault. Finally, if the system error status is reported when the fault is within a tolerable range for the mission mode, it is considered a false error report, which, in a real system would lead to safety system availability problems. A criterion is set up to distinguish between a detected fault and a false reporting based on the fault magnitude. Some fault models have two polarities. For example, a gain error can be either higher or lower than normal. Both polarities are simulated in this project. The fault injection process flow chart is depicted on Figure 6:

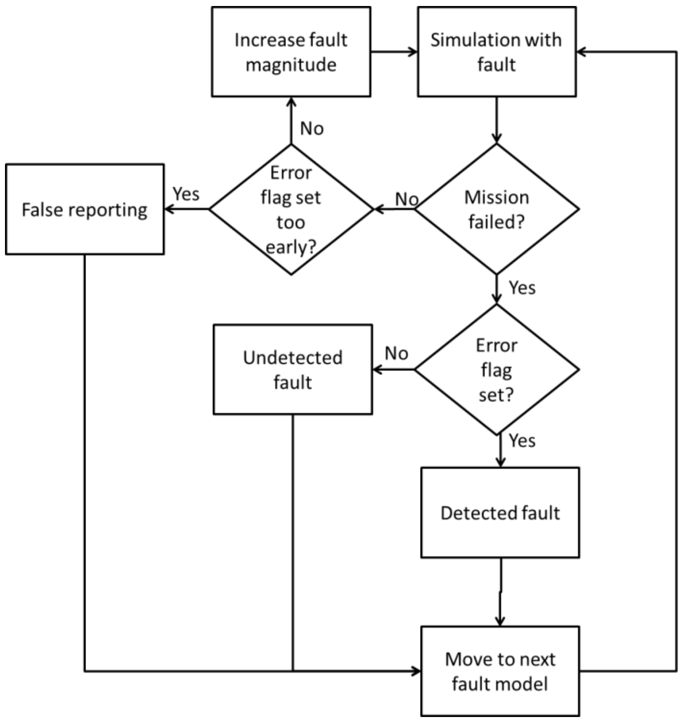


Figure 6. Fault injection simulation flow chart.

In this project, there are four analog blocks examined using fault injection method. There are five fault models implemented based on ISO 26262 recommendation [1]. Not all faults can be modeled accurately using this Simulink model. For some complex faults, such as incorrect output dynamic range, it would be better to use circuit level simulation for better accuracy.

III. RESULTS

The mission mode failure point is found for each fault model at each targeted analog block. Some fault models are only applicable to certain block. Two scenarios are simulated: a car at some distance away, and a child pedestrian at half the distance away. The smallest fault magnitude which causes either of the two scenarios to fail is recorded. The model does not provide the time required to report the error status to the master device. A full system simulation (with production RTL and firmware) is required to ensure that the error status is reported within the specified fault detection and handling time interval.

A total of 16 faults are simulated. The results are summarized in Table 1. 37% of the faults are detected before system failure (marked as D), another 37% are not detected (marked as ND), and the remaining 25% are detected only when the system is near the margin of the failure point (marked as M). Empty item indicates the particular fault model is not applicable to that analog block. There is no false reporting found in the simulations.

Table 1. Fault Injection Simulation Results
(ND = not detected, D = detected, M = marginal)

Analog Blocks / Fault Models (polarity)	Block A	Block B	Block C	Block D
Fault A(+)	ND		ND	ND
Fault A(-)	D		D	D
Fault B(+)	M		M	M
Fault B(-)	D		D	D
Fault C	ND			
Fault D		M		
Fault E(+)			ND	
Fault E(-)			ND	

Table 1 shows that the diagnostic mode has good coverage on faults with negative polarity. This means that the faults are detected at some point before mission failure. This is the desired coverage level because it makes sure all the unreliable measurements are detected and reported. On the other hand, faults with positive polarity are not detected before the mission failure point.

With this result available, the designer can modify the diagnostic mechanism accordingly. In this project, an additional diagnostic cycle is added. The Simulink model is modified and re-simulated with the new feature. The new feature increases the diagnostic coverage. 87% of the faults are detected before system failure. The remaining two undetected faults are situation dependent (object distance and input noise level). Table 2 shows the updated summary with the new diagnostic mechanism:

Table 2. Fault Injection Simulation Results with the New Diagnostic Mechanism

Analog Blocks / Fault Models (polarity)	Block A	Block B	Block C	Block D
Fault A(+)	D		D	D
Fault A(-)	D		D	D
Fault B(+)	D		D	D
Fault B(-)	D		D	D
Fault C	D			
Fault D		D		
Fault E(+)			ND	
Fault E(-)			ND	

IV. DISCUSSION

This paper demonstrates an alternative workflow for model-implemented fault injection in ASIC design. The biggest advantage of using Simulink as compared to circuit level model is time efficiency. With this workflow, the designers can focus on the functional level of the system. It is easier to implement the fault model because there are fewer components on the model. The simulation also runs faster as compared to mixed-signal simulation using circuit simulator such as Cadence Virtuoso. It takes 3 minutes to complete one simulation using this Simulink model. On the other hand, it takes hours, depending on the level of abstraction, to complete one run in mixed-signal simulation.

Because Simulink model is easier to create and modify, it can be a useful tool for feasibility study. For example in this project, designer can add the new diagnostic cycle and review the coverage quickly. It took about 2 man-hours to modify the model and update all the results. However, there are limitations on system level model-implemented fault injection. For example, structural faults on primitive components cannot be implemented in this level. A circuit level model is needed for structural fault injection. Also, using system level model-implemented fault injection can lead to inaccurate result. The result accuracy depends on the accuracy of the system model and the fault model. A validation method is needed in order to confirm the result.

V. FUTURE WORK

This is the first time we introduce Simulink modeling into our ASIC design workflow. There are improvements that can be made to maximize the benefit. First, an automated test bench could be created. It can iterate the fault model and fault magnitude to find the failure point and the diagnostic coverage following the flowchart on Figure 6. This approach reduces human involvement of running the simulation. The fault injection report can also be automatically generated when testing new design.

Second improvement that could be done is adding software fault model to the system. The DUT software can be described by blocks of functional model. Faults can be introduced to each block using the same methodology as discussed on the implementation section. Similar work has been demonstrated by research at KTH Industrial Engineering [4].

VI. REFERENCE

[1] ISO (2016) *Road vehicles — Functional safety Part 11: Guideline on application of ISO 26262 to semiconductors,* draft.

[2] University of Illinois at Urbana-Champaign (1997) *Fault Injection Techniques and Tools*, 75 – 82.

[3] Synopsys (2016) *Z01X Functional Safety Assurance*, 1 – 2.

[4] KTH Industrial Engineering and Management (2011), *Model-Implemented Fault Injection for Robustness Assessment*, 1 – 33.

[5] AEB VRU systems (2015) *European New Car Assessment Programme Test Protocol, 1.0.1*.