Holistic FMEDA-Driven Safety Design and Verification for Analog, Digital, and Mixed-Signal Design

By Robert Schweiger, Dominik Langen, and Jörg Müller, Cadence

With state-of-the-art electronics propelling the automotive industry into the future, automotive OEMs require safety-certified semiconductors. The integration of these advanced technologies into cars drives a need for component suppliers to assess and audit the risk of the technologies they want to deploy. At the same time, safety requirements are constantly evolving and becoming more stringent. To tackle these challenges, engineers are looking for a highly automated, integrated functional safety solution to help them achieve ISO 26262 certification faster.

With Cadence’s Failure Mode Effect and Diagnostic Analysis (FMEDA)-driven safety design and verification solution, integrating analog/mixed-signal and digital applications allows customers to ensure their automotive semiconductors meet rigorous safety standards while accelerating the ISO 26262 certification process. This white paper illustrates how to leverage the Cadence® Midas™ Safety Platform as part of an integrated safety flow to enable a FMEDA-driven safety methodology, including safety design, analysis and verification for analog, digital and mixed-signal design.

Contents

Introduction………………………………………………….2
Hardware Architectural Metrics…………………………2
Safety Mechanisms ..............................................3
The Safety Flow ................................................4
FMEDA-Driven Safety Methodology ..................4
Digital Functional and Safety Co-Verification ..6
Analog Safety Verification .................................10
Safety Implementation .................................13
Conclusion and Future ........................................14
Introduction

In recent years, safety design and verification for digital circuits have made great progress. However, more than 80% of field failures are due to the analog or mixed-signal portion of products [4]. Hence the safety methodology must be enhanced to enable an integrated safety flow for analog/mixed-signal and digital design. According to the ISO 26262 standard, this flow should include a FMEDA to provide good estimations early in the design cycle to guide the safety design and verification.

Safety cannot be an afterthought and must be addressed early on and at all stages of SoC development in safety-critical verticals, such as automotive, health, and aerospace/defense. Good safety practices start with a safety architecture phase, which helps to define the safety case and its requirements. A quantitative FMEDA can be used during the concept phase to analyze the hardware safety metrics using estimations for the area consumption of the different building blocks and for the effectiveness of the safety measures. It’s essential to identify failure modes of the hardware components and parts, calculate the probability of occurrence early in the design cycle, and define appropriate safety mechanisms to detect faults. Based on the predicted failure rates, the risk for a violation of a safety goal can be determined early in the design cycle and an appropriate safety architecture can be defined. Avoiding later architectural changes is key because these are very time- and effort-consuming loops and, in many cases, it is impossible to fix safety issues in later stages of the product development.

However, since this approach starts with predicted failure rates, all numbers should be verified to avoid either overdesign of the safety architecture or even worse underestimation of failure rates. Safety verification (simulation or formal) is a much more accurate approach to determine the diagnostic coverage (DC) and calculate the hardware safety metrics. The downside is that it is dependent on the availability of a hardware description, such as in RTL, gate-level, or transistors, and hence is deployed later in the SoC design process.

Hardware Architectural Metrics

The level of safety one should strive to reach when planning a safety-critical SoC varies depending on its functionality: a chip controlling air conditioning, for example, isn’t nearly as important for the safety of the vehicle than the system controlling the brakes.

Safety verification, as per the ISO 26262 standard, seeks to prevent two kinds of failures in automotive systems:

- Systematic, where the failure is deterministic and inherent in the design
- Random, where the failure is not deterministic and could be caused by usage conditions

This latter category contains both permanent faults, where a bit is stuck at one or zero, and transient faults, including single event upsets (SEUs) or other soft errors. In the following paragraphs, we will focus on random failures.

The ISO 26262 standard [7] defines four different automotive safety integrity levels (ASIL)—A, B, C, and D—where ASIL D represents the highest safety level. Depending on the ASIL level, the hardware architectural metrics should be calculated and fulfilled, including single-point fault metric (SPFM), latent fault metric (LFM), and probabilistic metric for hardware failure (PMHF), as shown in Table 1. The failures-in-time (FIT) rate is determined by the number of random failures that can be expected in one billion (10^9) device-hours of operation. The FIT rates for each safety-related element add up for the overall system if not mitigated by safety mechanisms.

<table>
<thead>
<tr>
<th>ASIL</th>
<th>SPFM</th>
<th>LFM</th>
<th>PMHF</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Not relevant</td>
<td>Not relevant</td>
<td>&lt; 1000 FIT</td>
</tr>
<tr>
<td>B</td>
<td>≥ 90%</td>
<td>≥ 60%</td>
<td>&lt; 100 FIT</td>
</tr>
<tr>
<td>C</td>
<td>≥ 97%</td>
<td>≥ 80%</td>
<td>&lt; 100 FIT</td>
</tr>
<tr>
<td>D</td>
<td>≥ 99%</td>
<td>≥ 90%</td>
<td>&lt; 10 FIT</td>
</tr>
</tbody>
</table>

*Table 1: Target values for hardware architectural metrics for each ASIL*

For more information on the basics of functional safety methodologies for automotive applications and ASIL compliance, refer to [2].
Safety Mechanisms

Safety mechanisms are the features on a chip that detect and mitigate or make the design tolerate faults and report them when faults have been detected. In order to work properly, safety mechanisms must be as reliable as possible, such as by keeping them inside a subsystem with an independent power and clock domain. This concept is referred to as a “safety island”—the physical area of a chip that is served by a given safety mechanism that manages all safety-related operations such as gathering of fault indication, fault management, and control of the system. However, safety mechanisms can also stretch across multiple domains—for example, a system-wide error correction code (ECC) protection works well over several domains and actually is intended to provide an uninterrupted chain of safety mechanisms while data is transported.

A special case of a digital safety mechanisms is a fault tolerant voting circuit driving a majority value decision of inputs to report a result. This is usually implemented as triple modular redundant system (TMR), which is more robust against data failures with increased fault coverage.

Another class of common digital safety mechanisms are ECCs to prevent bit errors. ECCs use redundancy in the transmitted data that can be used to recover the original message even if some bits are erroneously flipped. ECC, a subtype of cyclic redundancy check (CRC), assigns a checksum a given group of data that can be checked upon retrieval to ensure that no data corruption has occurred.

Error detection codes are widely used in data storage or in systems that interface between digital and analog contexts, as there is a high potential for accidental data loss. A good example is CRC, which covers packets transmitted via a serializer/deserializer (SerDes). In some cases, it’s just easier and cheaper to handle the faults in the digital domain compared to the analog domain, such as by using a CRC check, but there are also examples where safety mechanisms are built in analog circuits.

While safety mechanisms for digital circuits are widely used and accepted, analog components are mainly treated as a “safety black box” outside the safety verification process [6]. However, according to ISO 26262, the analog circuits must also be considered in the safety verification process. To do so, analog monitoring functions, such as voltage or clock generation, can be enhanced by analog safety mechanisms and integrated on a single chip to considerably simplify the safety analysis and verification.

In Figure 1, a low drop-out (LDO) regulator is the functional block and the regulated output voltage is the functional output to be monitored. A window comparator is the safety mechanism and monitors that the output voltage stays within the safe operating region. There are more safety mechanisms. The above examples should illustrate the key concepts for digital and analog. For more information on safety mechanisms, refer to [1].

Figure 1: Voltage regulator with analog safety mechanism using a window comparator monitoring the output
So how good are your safety mechanisms at detecting faults? Since each safety mechanism comes at a price, such as area overhead or performance impact, it’s important to carefully decide where to use which type of a safety mechanism. The effectiveness of a safety mechanism to detect faults and to prevent a safety goal violation can be measured by the DC metric. It can be determined by fault injection (FI) simulation, a widely accepted method recommended by ISO 26262 [7]. Each safety mechanism has certain capabilities to detect permanent and transient faults.

The residual failure rate $\lambda_r$ is

$$\lambda_r = \lambda (1 - \text{DC})$$

where $\lambda$ is the base failure rate of a part or subpart where a hardware “subpart” is defined as a logical division of a hardware “part”, such as an arithmetic logic unit in a CPU.

The DC of a safety mechanism for single point faults (SPFs) and latent point faults (LPFs) can be different.

**The Safety Flow**

In order to provide a holistic safety design and verification solution, Cadence has significantly enhanced its analog, mixed-signal, and digital flows.

The Midas Safety Platform (Figure 2) is a key technology to enable a FMEDA-driven safety methodology and flows. In addition, the Cadence Legato™ Reliability Solution, including an analog defect identification and simulation approach based on the concept of defect-oriented test, has been developed in alignment with the proposed IEEE P2427 standard [4, 9].

**FMEDA-Driven Safety Methodology**

A FMEDA is often the first step of a system safety study, which identifies potential failure modes and the consequences of those failures on different system levels.

An architectural FMEDA can be used to create the technical safety concept of the SoC providing early estimations on the safeness of the design at SoC level. Not only is this an ISO 26262 requirement, but it also ensures that the safety measures made in a design are adequate to reach the desired ASIL. If needed, the safety architecture can be still modified as it is early in the design cycle.
The problem with many FMEA and FMEDA tools is that they do not provide a direct link to commonly used IC design environments to benefit from additional native chip design data, such as design hierarchy and number of transistors. This is exactly where the Midas Safety Platform comes in, supporting different types of FMEDA.

If there is no native chip design data available, you can start with an early-phase functional safety exploration, leveraging the Midas Safety Platform for an architectural FMEDA. After creating the FMEDA hierarchy and defining and assigning the safety mechanisms, the safety concept can be analyzed by evaluating the hardware random failure metrics (SPFM, LFM, PMHF) based on early estimations of the area related to the different failure modes. However, since the FMEDA relies initially on estimations, which should be done rather conservatively, the achieved metrics need further validation.

A detailed FMEDA is a structured bottom-up approach to define failure modes, failure rates, and diagnostic capabilities of hardware components. It’s a validation of the architectural FMEDA according to the partitioning of hardware functions, failure modes, and failure rates to confirm and fine-tune the numbers. The FMEDA-based analysis evaluates the hardware random failure metrics and provides engineers guidance for the safety design, verification, and optimization of the hardware components and its subparts. It also helps to define the optimal number of safety mechanisms and its locations in the design in order to improve the DC.

If there is already a SoC design or parts of the design available, you can directly import the design hierarchy from the design database of the Cadence analog/mixed-signal and digital flows. In this case, you can map the imported design hierarchy to the FMEDA hierarchy, which can be different. Using the imported design data (RTL or netlists), already more accurate hardware random failure metrics can be calculated as the area (number of transistors) is now well determined and not estimated anymore.

Based on predicted failure rates, the risk for a violation of safety goals can be determined early in the design process. This step is pivotal to the success of any subsequent safety verification tasks. Safety verification methods, such as formal analysis or simulation, help to classify faults and hence to calculate more accurate DC values for the FMEDA.

One of the key advantages of the Midas Safety Platform, compared to other commercially available third-party FMEDA tools, is its direct access to the IC design database of the Cadence IC design flows. IC design data, such as the number of gates/flops or transistor count, area size, or design hierarchy structure, can be imported into the Midas Safety Platform to perform a more accurate and detailed safety analysis.

Figure 3: The Midas Safety Platform enables FMEDA-driven safety design, analysis, and verification planning
There are three key phases within the FMEDA process:

1. Architectural FMEDA: No design data available → early estimations
2. Detailed FMEDA: RTL or netlists available → base failure rates from design and estimated DC values
3. FMEDA validation: On RTL or netlists → simulation-based DC values

The Midas Safety Platform (Figure 3) helps to set up a FMEDA-based safety analysis and guides the engineer through all key safety steps of the FMEDA process: configuration, analysis, validation, and result.

The first step starts with the creation of the FMEDA hierarchy and running a first FMEDA based on early estimations. To do so, failure modes must be defined and mapped to the FMEDA hierarchy elements (parts, subparts). Next, estimated failure rates $\lambda_p$ should be defined by using area estimations for the different failure modes, DC targets set, and safety mechanisms assigned. Based on estimated failure rates, $\lambda_p$ and DC of the safety mechanisms, the residual failure rate $\lambda_{pr}$ for each failure mode can be determined and the SPFM and the LFM can be automatically calculated.

As design data becomes available, a design hierarchy extraction can be done to import the chip design from the IC design database and map it to the FMEDA hierarchy. With this formal link between the actual hardware partitioning and its design instances, the parts/subparts and failure modes represent a specific configuration for the subsequent safety analysis.

As the chip design progresses in parallel and more precise IC design data becomes available, such as an updated design hierarchy and more accurate numbers of devices and area, it can be directly leveraged to rerun the FMEDA analysis and recalculate the hardware safety metrics, thanks to the Midas Safety Platform’s direct access to chip design data.

Analysis results are now based on refined estimated failure rates but do not yet provide simulation-based DC values for the various safety mechanisms of the SoC. For FMEDA validation, a safety verification plan with the fault injection campaign and execution configurations for permanent and transient faults must be created. The goal of this validation step is to confirm the assumed DC values.

In order to set up a safety verification plan in the Midas Safety Platform, the observation, detection points, and test lists must be defined for all the failure modes. The safety verification plan can incorporate analog and digital data, design parts, failure mode descriptions, and safety mechanisms, and can drive the fault campaign scope for both flows during the safety verification. Further, the safety verification plan provides key information to configure the fault injection campaign in vManager Safety. At the end, the Midas Safety Platform automatically generates a detailed FMEDA report including the hardware architectural metrics.

**Digital Functional and Safety Co-Verification**

So, what’s the DC for each failure mode? This is where safety verification comes into play.

The idea of adopting functional safety requirements into an existing design or verification flow may seem daunting, but it’s not very different to verification at all. Many of the tools and features used in safety verification will seem familiar to verification engineers not accustomed to working on safety projects.

Since the Cadence safety verification flows and engines are based on the Cadence functional verification environment, both flows are fully compatible. This reduces the effort to develop and maintain different environments and enables the reuse of functional verification testbenches for the safety verification, which saves a lot of time for the fault campaign setup.

Cadence provides a unified functional verification environment called vManager Verification Management with vManager Safety for safety verification. Systematic failures such as design errors are mainly addressed by functional verification. Random failures are covered by safety verification.

vManager Safety controls all of Cadence’s verification engines, including the Xcelium™ Safety Simulator, Jasper™ Functional Safety Verification (FSV) App, and Spectre® Simulation Platform. A single database containing the results of all fault simulations for all digital verification engines unifies DC.
The purpose of safety verification is to classify faults in safe faults ($\lambda_s$), dangerous detected faults ($\lambda_{dd}$), and dangerous undetected faults ($\lambda_{du}$) to determine the DC, which is:

$$DC = \frac{\sum \lambda_{dd}}{\sum \lambda_{dd} + \sum \lambda_{du}} = \frac{\sum \lambda_{dd}}{\sum \lambda_d}$$

The safety verification plan created by the Midas Safety Platform guides the fault campaign scope for the analog and digital flows during the safety verification. vManager Safety (Figure 4) uses the safety verification plan as input to set up and manage fault injection campaigns. This includes the fault list generation, optimized coverage-based test selection, fault campaign scheduling, and collection of results. It can select and rank tests, extract and reduce a fault list, and control a simulation dynamically, giving the engineer full control of the fault campaign's progress.

In order to start the safety verification, vManager Safety schedules and executes the fault campaigns as defined in the safety verification plan and monitors their progress. Faults are detected by comparing a circuit's outputs at defined observation and detection points with and without fault injection. Functional outputs correspond to the observation points and the checker outputs of the safety mechanisms correspond to the detection points. Any mismatch in value or time causes the fault to be observed (functional output) or detected (checker output).

Fault classification based on simulation or formal methods helps to determine more accurate DC values, which can be back-annotated to the Midas Safety Platform to recalculate the hardware random failure metrics.

Focusing fault campaigns first on hardware components with the highest residual failure rates $\lambda_r$ can help to speed up the overall safety verification process and achieve the ASIL target faster.
As shown in Figure 5, vManager Safety has executed a fault campaign with 1248 possible faults, which resulted in 882 prime faults. 382 out of these 882 prime faults were classified as safe by the Jasper FSV App and hence there was no need to simulate those faults. As five different tests were used in this example, 2500 simulation runs were planned to be executed. Due to the test dropping feature of the functional safety flow and the pruning of test-based unobservable faults, only 2347 runs needed to be run in Xcelium Safety.

The test dropping feature removes faults from the list of faults to be simulated if the fault has already been detected in a previous simulation run. This feature can be controlled by the user. As a result of this fault campaign, 205 faults have been identified as “dangerous detected” (DD) and 382 faults as “safe” (S). 30 faults were not detected by the safety mechanism in this example and classified as “dangerous undetected” (DU). Figure 5 also shows that 631 faults are classified as “unobserved undetected” (UU), which usually require further analysis and can result from an insufficient test coverage.

Since safety verification based on simulation is a computational expensive task, it’s important to optimize the fault list and select the right verification engine for a specific test. For digital safety verification, the Xcelium Safety Simulator and Jasper FSV App complement each other and can be seamlessly used in a unified fault injection analysis flow.

There are different types of execution engines for fault classification available and a smart combination of those listed in Table 2 enables an effective functional safety verification.

<table>
<thead>
<tr>
<th>Safety Verification Engine</th>
<th>Use Case</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xcelium Safety Simulator</td>
<td>Simulation-based engine (serial and concurrent fault simulation)</td>
</tr>
<tr>
<td>Jasper FSV App</td>
<td>FSV structural and FSV formal fault analysis and fault list optimization</td>
</tr>
<tr>
<td>Spectre Simulator</td>
<td>Analog fault simulation using DFA and TFA, defect oriented test</td>
</tr>
<tr>
<td>Spectre AMS Designer</td>
<td>Mixed-signal fault propagation across boundaries between analog and digital blocks, ensures correct response time</td>
</tr>
</tbody>
</table>

Table 2: Safety verification engines and their use cases
Xcelium Safety injects faults on the nodes identified in the safety verification plan and provides the data for the metrics that engineers can use to identify potential issues. It supports serial fault simulation and a concurrent fault simulation mode. Serial fault simulation handles one fault per simulation run, whereas the concurrent fault simulation handles multiple faults in one simulation run. In this case, each gate retains a list of fault copies that store the status of a fault exhibiting the difference from fault-free values.

By performing a concurrent simulation of the good and faulty versions of the circuits, Xcelium Safety can save repeated simulations of the good parts of the circuit, meaning parts of the circuits that are not affected by a given fault. The concurrent fault simulation can significantly reduce the runtime of a fault simulation campaign. The high simulation speed of the concurrent simulator comes at the price that not all language constructs of the hardware description languages are currently supported. While gate-level designs can usually be simulated completely by the concurrent fault simulator, this is not entirely true for RTL designs. If the concurrent fault simulator cannot simulate a fault, the flow will automatically direct this fault to the serial fault simulator, which can be used for any design.

The Jasper FSV App complements the Cadence functional safety flow by adding formal fault qualification and propagation analysis to assist and improve overall safety analysis. The formal verification tool supports structural (FSV structural) and formal (FSV formal) fault analysis techniques, which are used at different verification phases in the flow to verify if the injected faults affect the results at one of the diagnostic points.

Structural analysis slots into the Xcelium Safety Flow just before Xcelium Safety itself (Figure 6), where it can be deployed to analyze the connectivity, activatability, and propagatability of faults and to reduce the number of fault simulations in Xcelium Safety by optimizing the fault list.

With connectivity analysis, it can be proven that a fault has no physical connection to the functional strobe point and is outside the cone of influence. Hence it is considered as a safe fault.

Activatability analysis checks whether the fault can be functionally activated from the inputs. If the fault cannot be activated due to a propagated constant the fault is considered as safe.

For all remaining faults, the propagability analysis (the third category of safe faults) identifies barriers in the design that prevent the fault from reaching the observation points (unobservable).

In addition, fault relation analysis can be used to extract fault relations such as equivalence or dominance of faults. If two faults produce the same results for any input stimulus (equivalence), faults are statically reduced. If a fault A is observed for any particular input stimulus, fault B is also observed (dominance), then the fault is conditionally reduced. As a result, fault pairs can be collapsed and only one representative fault (prime fault) should be simulated.

The Jasper FSV App's structural analysis executes those analysis types independent of the tests executed and automatically annotates structurally safe faults in the common fault database (untestable (UT)). These faults don’t need to be simulated.

The Jasper FSV App's test constants executes a subset of those analysis check types again, but now considers the constants extracted from the simulation of the tests to be used for the fault injection simulations. It can predict which faults won’t be observable while running a certain test (test-based unobservable (UU)). Because these faults can be removed from the list of faults to be simulated, this step is called fault pruning.

The advantage of formal verification using the Jasper FSV App is that it is based on mathematical methods, is exhaustive, and requires no testbench, thus saving months of verification effort and increasing design quality by finding more bugs at an earlier stage compared to other verification methods. Because formal tools have certain capacity limits in terms of number of gates and faults, it is usually applied after fault simulation to identify which of the remaining unobserved faults logged in the database are formally safe.
The Jasper FSV App’s formal verification leverages this power of formal engines and can find additional safe faults or create example propagatability traces for undetected faults. Those traces can be translated to additional tests or STL segments for the fault simulation or a STL library, and Xcelium Safety can harvest the additional fault coverage with them.

Applying these methods on a typical example design achieved a reduction of 29.5% of the number of faults to be simulated [5]. In general, the reduction can vary between 0% and 85% depending on the strobes and other factors of a particular fault campaign.

Safety verification may sound like security verification, and they are indeed related, but safety verification is primarily focused on faults caused by non-malicious actions. Security verification doesn’t include safety concerns by itself, but ASIL-D systems must often address security concerns to ensure compliance. The Jasper FSV App’s formal verification also offers advanced check types and fault models to perform a formal custom safety and security analysis. The goal of a security vulnerability analysis is to check the ability to detect or eliminate hacker attacks in secure subsystems.

**Analog Safety Verification**

While analog and mixed-signal components are the main source of test escapes that result in field failures, a lack of tools to analyze design test coverage has made it difficult for designers to address the issue. The IEEE P2427 Working Group has standardized the definitions of manufacturing defects for analog simulation. In order to analyze the effects of manufacturing defects on the circuit, defect models such as DC short, DC open, AC coupling, resistive bridge (short/open), and others have been defined. This methodology for analog defect simulation to analyze the analog test coverage was a prerequisite.

Cadence has developed the Legato Reliability Solution, which is built on the Cadence Virtuoso® design platform, and Spectre Simulator to automate fault simulation and to determine analog test coverage [4] through functional pattern. Existing Virtuoso users can incorporate analog defect analysis into their methodology with little change to their flow (Figure 7).

The analog goals are similar to the digital ones in that the test coverage must to be determined. After the FMEDA, the Midas Safety Platform passes the safety verification plan to vManager Safety to set up the fault campaigns as described earlier. Virtuoso ADE Assembler is leveraged to identify the faults, set up the specific tests, and define the simulation run mode for the analog fault simulation for each test.
The methodology to determine the test coverage is a three-step process comprised of defect identification, fault simulation, and post-simulation analysis:

1. **Defect Identification**: The Fault Assistant tool in Virtuoso ADE Assembler automatically generates the defect list for the fault simulation.

2. **Fault Simulation**: Spectre Simulator supports analog fault simulation modes and can be used as an analog solver within Spectre AMS Designer to enable analog/mixed-signal fault simulations.

3. **Post-Simulation Analysis**: Analyzes overall test coverage and effectiveness and debugs test issues.

In addition to the Fault Assistant, Cadence also enhanced Virtuoso ADE Assembler to support and optimize analog defect simulations. Managing a large number of faults can be challenging, so there are several approaches available to designers, including fault collapsing, fault weighting, and advanced fault sampling techniques.

The goal of the analog defect simulation is to maximize the detection coverage and minimize the testing time. Spectre Simulator supports now two dedicated simulation modes called direct fault analysis (DFA, accurate mode) and transient fault analysis (TFA, fast mode).

In the DFA mode of Spectre Accellerated Parallel Simulator (APS), the faults are injected into the nominal design at the beginning of the simulation analysis, such as DC operating point, transient, or AC small signal. It provides the most accurate results and the most complete description of the defect’s effect on the circuit’s performance. This mode also serves as a reference to verify the effect of defects on the circuits. For a large number of faults, Spectre APS DFA mode can be also used in a distributed simulation mode.

The TFA mode in Spectre APS provides several methods for accelerating defect simulation. It provides different defect simulation modes to reduce simulation time. Designers can typically start with the fastest method and change then gradually to more accurate methods until all defects are detected. The TFA mode can typically achieve a total speed up of 20X compared to DFA [4].

In addition, Spectre AMS Designer can be used to enable analog/mixed-signal fault analysis to further improve the fault simulation performance using different engines. Of course, simulation runs can be also distributed on compute farms for maximum throughput.
As shown in Figure 8, the Virtuoso ADE Assembler Fault Simulation mode displays the results of the defect simulation and the coverage by test including the overall coverage. In this example, the test “DCGain” detected 40 faults and just one fault was undetected, which is a test coverage of 97.56%. The test “Swing” detected 19 out of 41 faults, a coverage of only 46.34%.

The defect detection matrix provides graphical information on which test pattern achieves the best coverage for identifying a fault. The first column in Figure 9 lists all injected faults. Each subsequent column represents a different test. Tests that show many detected faults (D, marked in red) in the column provide good coverage. Tests with lots of undetected faults (U, marked in green) indicate that the test coverage is low. If a row in the defect detection matrix is completely green, it means that the corresponding fault could not be detected at all.

Because the Legato Reliability Solution’s analog defect simulation is based on the Virtuoso ADE Assembler, cross-probing from the report back into the Virtuoso Schematic Editor for defect visualization and debugging is also supported.

Once the analog defect simulation is completed, the verification results in the Legato Reliability Solution are reported back into vManager Safety to track the overall safety verification requirements and progress, completing a cycle that closely matches functional verification.
Safety Implementation

Of course, a holistic safety methodology doesn’t stop with the safety verification but also requires a safety-aware implementation approach to meet the functional safety standards. The scope of this last section is to show the effort made by Cadence to create an automated safety-aware digital implementation flow with a tight connection between logic synthesis, test, P&R, and equivalence checking tools. The idea is to provide a full set of tools working together in an integrated flow (Figure 10) and pass safety requirements from Genus Synthesis to Innovus Implementation P&R to provide a faster path to implementation to achieve automotive system-on-chip (SoC) safety, quality, and reliability targets.

Safety-critical automotive designs require some special implementation features to enable automated safety-aware P&R, including triple voting flop insertion, safety island generation, logic isolation buffer insertion, and specification of different routing constraints. By leveraging a fully integrated flow, digital safety mechanisms can be designed, managed, and verified with Genus Synthesis, Innovus Implementation (P&R), Cadence Modus DFT Software, and Conformal Equivalence Checker.

Genus Synthesis can automatically insert safety mechanisms, such as TMR flops or dual-core lock-step (DCLS) controllers, and speed up the synthesis of safety mechanisms. Innovus Implementation can then automatically create and control voting flop insertion and manage the creation, shape, and routing of the safety islands for the DCLS controller. The Cadence Modus DFT Software Solution can improve coverage of untested faults by identifying which are deterministic or use inactive logic and can report coverage pre-silicon. The Conformal Equivalence Checker can be used for signoff to ensure that the logic, including the dedicated safety structures implemented with Innovus Implementation, are functionally equivalent with the original RTL code synthesized by Genus Synthesis. The Midas Safety Platform has full access to the physical chip design data at any time to recalculate the FMEDA.

While there is a fully automated RTL-to-GDSII flow available for digital safety design and implementation, an analog and mixed-signal design flow does not provide this level of automation for analog synthesis and P&R for physical implementation.
Conclusion and Future

In this white paper, we have introduced a holistic FMEDA-driven safety design and verification methodology for analog, mixed-signal, and digital designs to address ISO 26262 requirements. Since all Cadence flows are tightly connected to the Midas Safety Platform, a comprehensive FMEDA-based methodology from the architecture phase to IC implementation has been enabled to accelerate the safety design, verification, and implementation.

Due to the significant amount of field failures caused by the analog or mixed-signal portion of products, a integrated analog fault simulation approach based on the concept of defect-oriented test was introduced to measure and maximize the detection coverage. The IEEE P2427 Working Group has standardized the definitions of manufacturing defects for analog simulations and has proposed a set of defect models, which significantly eases defect modeling and simulation.

Recently the Accellera Systems Initiative has formed a working group of functional safety industry experts to develop a standard that will provide a definition of functional safety data—the set of data needed to perform safety activities and to generate work products—exchanged to improve automation, interoperability, traceability, and retargeting. The standard will specify a data model, language, or format to exchange data seamlessly within functional safety-aware design flows and even the supply chain [8].

As a member of the IEEE and Accellera Systems Initiative, Cadence actively contributes to these standards with the goal to enable a faster certification of safety-critical automotive and industrial designs.

Reference


