Functional Safety Methodologies for Automotive Applications

Alessandra Nardi
Cadence Design Systems
anardi@cadence.com

Antonino Armato
Cadence Design Systems
armato@cadence.com

ABSTRACT

Safety-critical automotive applications have stringent demands for functional safety and reliability. Traditionally, functional safety requirements have been managed by car manufacturers and system providers. However, with the increasing complexity of electronics involved, the responsibility of addressing functional safety is now propagating through the supply chain to semiconductor companies and design tool providers. This paper introduces some basic concepts of functional safety analysis and optimization and shows the bridge with the tradition design flow. Considerations are presented on how design methodologies are capturing and addressing the new safety metrics.

KEYWORDS

Functional Safety, EDA (Electronic Design Automation), ISO26262, Automotive, Redundancy, BIST (Built-In-Self-Test), FMEDA (Failure Mode Effects and Diagnostic Analysis), DFA (Dependent Failure Analysis)

1 INTRODUCTION

The term "*automobile*" combines the Greek "*autòs*" (self, independent) and the Latin *mobilis* (moveable, mobile), where therefore auto-mobile means to be self-mobile [1]. While it might be up to debate whether the self was indicating the human or the car, today's era of smart, connected, self-driving vehicles surely brings to memory the word's original meaning.
The race to self-driving cars is making the news almost daily. This new market has been an incredibly fast driver for the evolution of SoC development for automotive applications. Advanced Driver Assistance Systems (ADAS), the precursor of fully autonomous vehicles, have led to an exponential increase in the amount and complexity of electronics in cars [2]. Modern luxury cars are reported to have up to 90 ECUs (Electronic Control Unit) [3] implementing several of the advanced features, such as Adaptive Cruise Control, Collision Avoidance System, Automatic Parking. ADAS applications require environment recognition based on processing of data from radar, lidar and camera and sensor fusion, which is very computationally intensive and require the support of advanced process nodes to meet the performance/watt demands. As a consequence, the automotive industry is also witnessing a migration to advanced technologies, which can present a bigger challenge for reliability (for example, process variation, electrostatic discharge, electromigration).

Therefore, safety becomes a fundamental requirement in the automotive systems to guarantee a tolerable level of risk. Safety can be defined referring to two existing safety standards: IEC 61508 [4], which is a functional safety standard for the general electronics market developed by the International Electrotechnical Commission (IEC), and ISO 26262 [5], which is a functional safety standard for automobiles developed by ISO. Introduced in November 2011, ISO 26262 has rapidly affirmed as the guideline for the automotive engineer [3].

Compliance to these requirements has been traditionally addressed by car manufacturers and system suppliers. However, with the increasing complexity, the industry is taking a divide and conquer approach, and all participants of the supply chain are now called to support and enable functional safety and reliability standards. These metrics are becoming an integral part of the semiconductor design flow.

This paper provides a brief overview of functional safety for automotive applications and ties it to the traditional EDA domain. Section 2 starts with an introduction to some functional safety concepts as defined in the ISO 26262 standard to address random and systematic failures: ASIL (Automotive Safety Integrity Level), failure classification and hardware metrics. Section 3 details functional safety analysis and Section 4 describes the functional safety flow for semiconductors and how it integrates with the traditional design and verification flow.

2 BASICS OF FUNCTIONAL SAFETY

In this Section, we review the basic concepts of Functional Safety, its lifecycle and the analysis techniques.

2.1 Definition of Functional Safety

ISO 26262 "Road vehicles – Functional Safety" is the automotive industry standard, derivative of the more general IEC 61508 functional safety standard [4], designed for safety-related systems for series production passenger vehicles with a maximum gross vehicle mass up to 3,500 kg and that are equipped with one or more E/E subsystems [7].

Based on ISO26262, Functional Safety is the "absence of unreasonable risk due to hazards caused by malfunctioning behavior of Electrical/Electronic systems".

This dry contractual definition can be represented as a chain of implications: Malfunction (of the E/E component) → Hazard (unintended situation) → Risk (of harm/damage) → Required risk reduction (based on acceptable level of risk).

The rest of this Section describes the elements of this chain in more detail.

2.2. Failure Classification and Hardware Random Failure Metrics

Per ISO26262, malfunction of the E/E component is classified into two types of failures:

- **Systematic failures:** They represent the failures in an item/function that are induced in a deterministic way during development, manufacturing or maintenance (process issues).
- **Random failures**: Hardware random failures appear during the lifetime of a hardware element and emanate from random defects innate to the process or usage conditions.

Hardware random failures can be further classified in permanent faults (e.g., stuck-at faults) and transient faults (e.g., Single-Event-Upsets or soft errors).

The systematic failures, that typically are due to process causes, can be addressed by a change of the design or of the manufacturing process, operational procedures, documentation or other relevant factors. Typical requirements are tracking and traceability. All these methods and expectation are captured by the Functional Safety Management activities as reported in ISO26262-2:2011.

Handling of random failures is addressed during the design and verification of the HW/SW system by introducing safety mechanisms to make the architecture able to detect/correct the malfunctions. From the vocabulary in ISO 26262:1-2011, a **Safety Mechanism** is technical solution implemented by E/E functions or elements, or by other technologies, to detect faults or control failures in order to achieve or maintain a safe state. Examples of safety mechanisms are ECC (Error Correction Code), CRC (Cyclic Redundant Check), hardware redundancy, BIST (Built-In-Self-Test) and several others. The effectiveness of the solution to detect these random failures is measured by three metrics: **Single Point Fault Metric (SPFM), Latent Fault Metric (LFM),** and **Probabilistic Metrics for Hardware Failures (PMHF).**

In a nutshell, SPFM and LFM are ratio metrics that measure the ability of the hardware component to detect faults, while PMHF is expressed in FIT (Failure In Time) and summarize the overall likelihood of risk to happen. These metrics are essentially the measurement of functional safety for hardware components per ISO 26262 and the rest of the paper mainly focuses on how to analyze them and meet their target value.

The description and formulas that define the **hardware architectural metrics** are reported in ISO 26262-5:2011, Annex D, C.2 and C.3 and 9.2:

- **Single Point Fault Metric**: Reflects the robustness of an item/function to the single point faults either by design or by coverage from safety procedures
- **Latent Fault Metric**: Reflects the robustness of an item/ function against latent faults either by design (primarily safe faults), fault coverage via safety procedures, or by the driver's recognition of a fault's existence before the infraction of a safety objective

Probabilistic metric of hardware failures: Provides rationale that the residual risk of a safety goal violation, due to random hardware failures, is sufficiently low [8].

In an intuitive way, a single point fault can lead directly to the violation of a safety goal, while a latent fault in an undetected fault which allows another fault to cause a hazard.

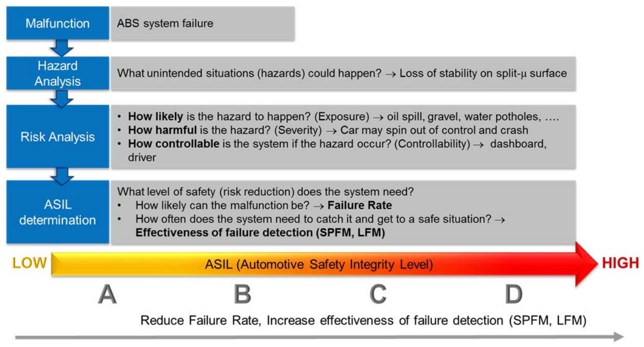

Malfunction
ABS system failure
Hazard Analysis
What unintended situations (hazards) could happen? → Loss of stability on split-μ surface
Risk Analysis
• How likely is the hazard to happen? (Exposure) → oil spill, gravel, water potholes, ….
• How harmful is the hazard? (Severity) → Car may spin out of control and crash
• How controllable is the system if the hazard occur? (Controllability) → dashboard, driver
ASIL determination
What level of safety (risk reduction) does the system need?
• How likely can the malfunction be? → Failure Rate
• How often does the system need to catch it and get to a safe situation? → Effectiveness of failure detection (SPFM, LFM)
LOW
ASIL (Automotive Safety Integrity Level)
HIGH
A
B
C
D
Reduce Failure Rate, Increase effectiveness of failure detection (SPFM, LFM)

Figure 1: ABS example of ASIL determination based on Hazard and Risk Analysis at the concept phase

2.3 ASIL (Automotive Safety Integrity Level)

Given a malfunction of a defined function at the vehicle level (e.g. ABS, Anti-lock Braking System), a hazard and risk analysis follows to determine the risk of harm/injury to people and the damage to property. This analysis is based on the Exposure, Severity and Controllability of the hazard and the resulting risk, and determines the **ASIL (Automotive Safety Integrity Level)**, i.e. the level of risk reduction needed to achieve a tolerable risk. Figure 1 shows an example of the steps that leads to the ASIL determination based on the malfunction and its potential impact. ASIL A is the least stringent level of safety reduction, while ASIL D is the most severe.

Table 1: **Failure metrics for each ASIL level**

ASIL	Failure Rate	SPFM	LFM
A	< 1000 FIT	Not relevant	Not Relevant
B	< 100 FIT	≥ 90%	≥ 60%
C	< 100 FIT	≥ 97%	≥ 80%
D	< 10 FIT	≥ 99%	≥ 90%

For hardware components, the ASIL requirements determines the values to achieve for the failure metrics as shown in Table 1. For addressing systematic failures, the ASIL will also set the strictness of process compliance (e.g. traceability, process quality, documentation).

2.4 Functional Safety Lifecycle and Development Phases

All the phases of the functional safety lifecycles are defined and documented in the ISO 26262 standard. Figure 2a illustrates the sequence of the concept and the development phases, while Figure 2b and 3c reports the corresponding Functional Safety activities with examples. The concept phase is owned by car manufacturers and defines the systems to implement a function at the vehicle level (called the "item" in ISO26262 terminology), e.g. the Automatic Emergency Braking System. The ASIL is determined at this level and the safety goals and the functional safety requirements are defined from it. When the system level product development phase starts, for each functional safety requirements, the technical safety requirements are derived with respect to the HW/SW components of the safety-related function.

Essentially, the safety goals start at the vehicle level and are mapped and refined during the development chain till the hardware failure metrics are defined and allocated to the various hardware subsystems.

In this paper, we are interested in the quantitative hardware architectural metrics of random failure and how to incorporate/evaluate safety mechanisms to match the requirements.

3 FUNCTIONAL SAFETY ANALYSIS

Functional Safety Analysis is used to evaluate the safety level achieved by the product (e.g. an IP, an SoC). It comprises of quantitative evaluations such as **FMEDA (Failure Mode Effect and Diagnostic Analysis)** and **Timing Analysis**, and qualitative assessments such as **DFA (Dependent Failure Analysis)**. These are described below.

3.1 FMEDA (Failure Mode Effect and Diagnostic Analysis)

FMEDA is a structured approach to define failure modes, failure rate and diagnostic capabilities of a hardware component.

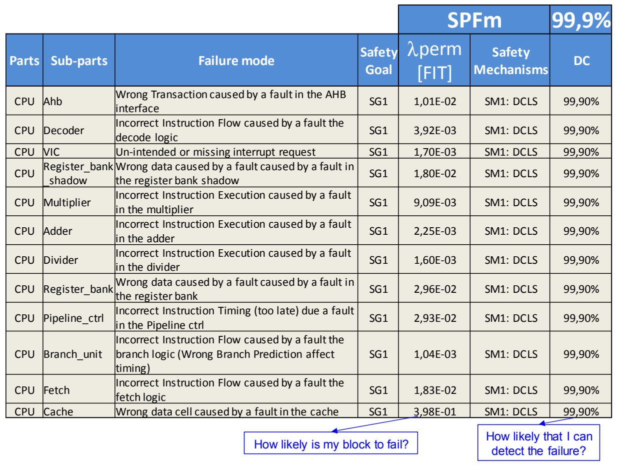
Based on the component functionality, the FMEDA hierarchy is structured in parts/subparts/elementary subparts (depending on the detail level)/failure modes [5]. Each failure mode is categorized as to whether it affects the safety goal or not.

For each failure mode defined and affecting safety goals, basic needed inputs will be:

- Failure Rate (FR): the rate at which the component experiences faults, i.e. the reliability
- Safety Mechanism (SM): whether there is a safety mechanism to detect the failure mode
- Diagnostic Coverage (DC): the effectiveness of the safety mechanism at detecting faults

The outputs are the hardware architectural metrics SPFM, LFM, and PHFM to assess the level of functional safety readiness.

Intuitively, these metrics capture how reliable the component is (how soon it is likely to fail) and how good the safety mechanism is at detecting the failure and bringing the system to a safe state.



				SPFm		99,9%
Parts	**Sub-parts**	**Failure mode**	**Safety Goal**	**λperm [FIT]**	**Safety Mechanisms**	**DC**
CPU	Ahb	Wrong Transaction caused by a fault in the AHB interface	SG1	1,01E-02	SM1: DCLS	99,90%
CPU	Decoder	Incorrect Instruction Flow caused by a fault the decode logic	SG1	3,92E-03	SM1: DCLS	99,90%
CPU	VIC	Un-intended or missing interrupt request	SG1	1,70E-03	SM1: DCLS	99,90%
CPU	Register_bank_shadow	Wrong data caused by a fault caused by a fault in the register bank shadow	SG1	1,80E-02	SM1: DCLS	99,90%
CPU	Multiplier	Incorrect Instruction Execution caused by a fault in the multiplier	SG1	9,09E-03	SM1: DCLS	99,90%
CPU	Adder	Incorrect Instruction Execution caused by a fault in the adder	SG1	2,25E-03	SM1: DCLS	99,90%
CPU	Divider	Incorrect Instruction Execution caused by a fault in the divider	SG1	1,60E-03	SM1: DCLS	99,90%
CPU	Register_bank	Wrong data caused by a fault caused by a fault in the register bank	SG1	2,96E-02	SM1: DCLS	99,90%
CPU	Pipeline_ctrl	Incorrect Instruction Timing (too late) due a fault in the Pipeline ctrl	SG1	2,93E-02	SM1: DCLS	99,90%
CPU	Branch_unit	Incorrect Instruction Flow caused by a fault the branch logic (Wrong Branch Prediction affect timing)	SG1	1,04E-03	SM1: DCLS	99,90%
CPU	Fetch	Incorrect Instruction Flow caused by a fault the fetch logic	SG1	1,83E-02	SM1: DCLS	99,90%
CPU	Cache	Wrong data cell caused by a fault in the cache	SG1	3,98E-01	SM1: DCLS	99,90%

How likely is my block to fail?
How likely that I can detect the failure?

Figure 3: Simplified FMEDA table example referred to safety goal SG1 covered by Dual-Core-Lock-Step (DCLS)

The failure rate is the measure of the reliability of a component, which is expressed in FIT. The FIT rate of a component is the number of failures expected in one billion hours of operation. In other terms, a device has a FIT rate equal to 1, has a MTTF (Mean Time To Failure) of 1 billion hours [5].

Per ISO 26262 the estimated failure rates for hardware parts shall be determined either [5]

- Estimated by application of industry reliability data books (e.g. IEC 61709, IEC TR 62380);
- Derived from observation of field incidents, such as analysis of material returned as field failures;
- Derived from experimental testing.

Figure 3 shows an example of a simplified FMEDA table.

For each failure mode, the FR, SM and DC are combined to calculate the SPFM, the LFM and the FIT rate. The total metrics are obtained by summation of all the rows. By analyzing the overall metrics and the row-by-row contribution, the FMEDA directs the designer to which parts of the design needs to be enhanced for safety readiness. The process of analyzing-designing-verifying to

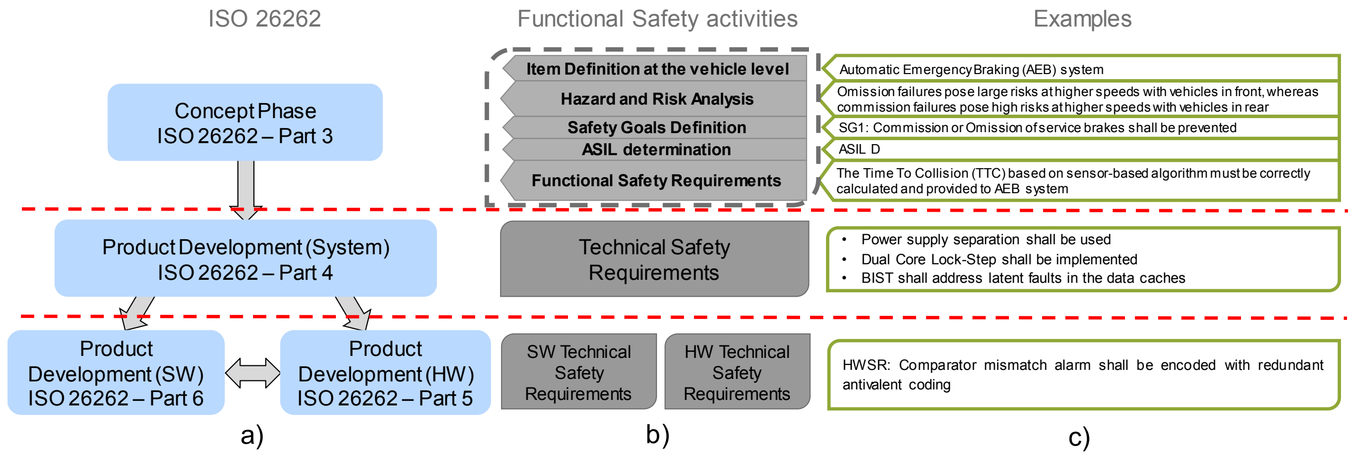

ISO 26262
Functional Safety activities
Examples
Concept Phase
ISO 26262 – Part 3
Item Definition at the vehicle level
Hazard and Risk Analysis
Safety Goals Definition
ASIL determination
Functional Safety Requirements
Automatic Emergency Braking (AEB) system
Omission failures pose large risks at higher speeds with vehicles in front, whereas commission failures pose high risks at higher speeds with vehicles in rear
SG1: Commission or Omission of service brakes shall be prevented
ASIL D
The Time To Collision (TTC) based on sensor-based algorithm must be correctly calculated and provided to AEB system
Product Development (System)
ISO 26262 – Part 4
Technical Safety Requirements
Power supply separation shall be used
Dual Core Lock-Step shall be implemented
BIST shall address latent faults in the data caches
Product Development (SW)
ISO 26262 – Part 6
Product Development (HW)
ISO 26262 – Part 5
SW Technical Safety Requirements
HW Technical Safety Requirements
HWSR: Comparator mismatch alarm shall be encoded with redundant antivalent coding
a)
b)
c)

Figure 2: Phases of the Functional Safety development process, corresponding requirements and examples

reach the target hardware metrics values is the focus of Section 4 and it essentially shows how Functional Safety plays a role in the traditional design flow.

In the specific example of Figure 4, the FMEDA has been performed for permanent faults. In a similar way, it is possible to build the analysis for transient fault.

3.2 Timing Analysis

Though the hardware architectural metrics described so far do not include timing constraints, it is easy to understand that the complete evaluation of the safety mechanisms shall involve timing performance. In fact, the system must be able to detect faults and transition to a safe state within a specific time (FTTI – Fault Tolerant Time Interval), otherwise the fault can become a system level hazard. This is captured in Figure 4, which also illustrates the Diagnostic Test Interval (DTI), the part of FTTI allotted to detect the fault [5]. Just as a reference example, the DTI for fault detection in a CPU can be around 10ms, while around 100ms would be allocated for FTTI of the whole system.

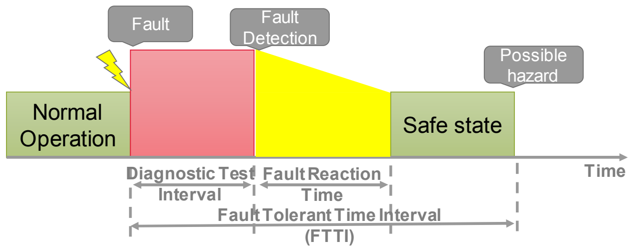


Figure 4: **Fault Tolerant Time Interval and Diagnostic Test Interval**

3.3 DFA (Dependent Failure Analysis)

Together with the analysis of hardware random failures, another aspect to evaluate, especially when the system has shared resources, is the **Dependent Failure Analysis (DFA)**.

The analysis of dependent failures, also known as analysis of possible common cause and cascading failures between given elements, aims to identify the single causes that could bypass a required independence or freedom from interference between given elements and violate a safety requirement or a safety goal [5].

Two most intuitive scenarios associated with architectural features are:

- Similar and dissimilar redundant elements
- Different functions implemented with identical software or hardware elements.

ISO 26262 provides a list of **Dependent Failure Initiators** to evaluate and guidelines on safety measures to control or mitigate this kind of faults.

Examples of Dependent Failure Initiators due to random hardware faults of shared resources are failures in clock elements, power supply elements or common reset logic, while Dependent Failure Initiators associated with random physical root causes are for example short circuits, latch up and crosstalk.

Typical countermeasures for random physical root causes are described in [5]:

- Dedicated independent monitoring of shared resources (e.g. clock monitoring)
- Self-tests at start-up (e.g. safety mechanism enabling check)
- Diversification of impact (e.g. clock delay between master & checker core)
- Indirect monitoring using special sensors (e.g. delay lines used as common-cause failure sensors)
- Fault avoidance measures (e.g. physical separation/isolation)

4 FUNCTIONAL SAFETY REQUIREMENTS DRIVING THE TRADITIONAL DESIGN FLOW

In essence, functional safety is all about inserting active measures for achieving the required risk reduction and there are two knobs to play with: reliability and active safety. This paper focuses only on how functional safety is deployed in the traditional RTL-to-GDS flow. In this Section, we will review how Functional Safety analysis is used in the hardware design/verification flow to achieve the required risk reduction, i.e. the required hardware safety metrics.

FMEDA drives design exploration to meet the functional safety targets. In fact, by looking at failure modes and their metrics, it tells where to focus the design effort for meeting the constraints. It also directs the fault injection campaigns to get a direct, more accurate evaluation of the safety mechanism diagnostic coverage. DFA is instead performed to review that proper measures are taken during the RTL-to-GDS flow to guarantee independence and avoid common-cause failures.

The selection of the best safety mechanism for a specific building block needs careful analysis of the tradeoffs between effectiveness and cost: power consumptions, area, safety metrics and timing performance all need to be evaluated.

Safety mechanisms can be software tests implemented anyway in the software stack, or hardware tests which are manually crafted into the RTL or automatically inserted through the design flow. This paper only focus on the latter.

4.1 Design and Implementation

The most notable example of safety mechanism already automated in the design flow is the built-in self-test (BIST), used for automotive in-system/field testing for lifetime reliability to achieve the desired ASIL.

There are two general categories of BIST techniques for testing random logic [9]. They have different impact on the safety metrics and require different timing performance:

- **Online BIST**: performed when the functional circuitry is in normal operational mode (mission mode). It contributes to the SPFM metric and has more stringent timing requirements since it must complete within the DTI.

- **Offline BIST**: performed when the functional circuitry is not in normal mode, e.g. during power-on reset at the engine startup. It contributes to the LFM and timing requirements are more relaxed.

Challenges to be addressed during BIST integration are speed testing capabilities, power consumption, area, routing minimization and ASIL target [9]. The integration of compression techniques provides quality and efficient sharing of resources [11].

It is worthwhile mentioning that, although correlated, the test coverage estimated during BIST insertion is not exactly the DC required by the ISO26262 metrics. As it will be explained in Section 4.2, Functional Safety verification might be needed to accurately measure the DC. Referring to the Automatic Emergency Braking system example in Figure 5, BIST can be used to avoid accumulation faults in the cache of a CPU (typical issue in complex microprocessors).

Another example of safety mechanism is the Triple Modular Redundancy (TMR) technique. In this case, the logic (memory cell) sensitive to Single-Event-Upsets is tripled and voters are placed at the outputs to identify the correct value [12]. Figure 6 shows a Triple Modular Redundancy architecture applied at the flip-flop level: this technique covers both SPFM and LFM for the sequential elements that are triplicated.

Whenever the safety architecture is based on hardware redundancy, DFA needs to be performed to address common-cause failure due to random physical root causes: essentially, logical independence needs to translate into physical independence.

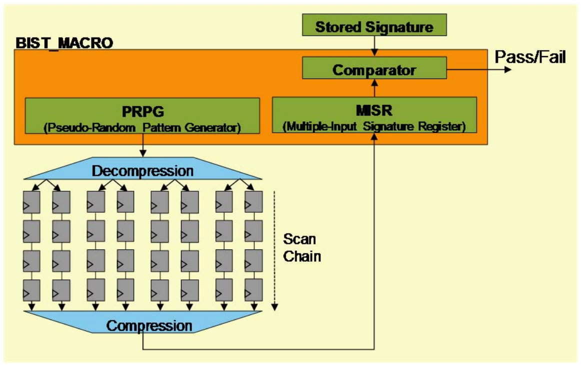


Figure 5: Example of BIST architecture

Another safety mechanism based on redundancy is the Dual-Core-Lock-Step (DCLS), mentioned in the example in Figure 2. Both shared resources (e.g. power supply, clock, reset signal) and single physical root cause need to be considered as potential common cause failures and require special design techniques to keep a high achievable DC.

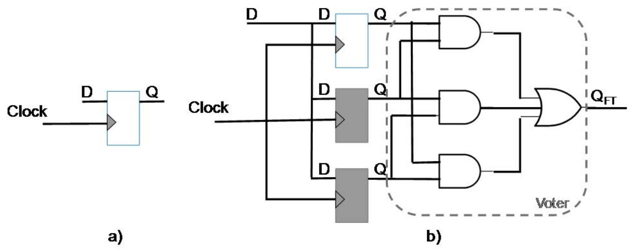


Figure 6: Triple Modular Redundancy (TRM) architecture applied to flip-flops

Figure 7 reports examples of countermeasures to address the common cause failure at the functional level, such as timing diversity and outputs inversion. It also shows layout techniques avoid cross-talk or guarantee strong isolation between redundant blocks (e.g. ring barrier). Several place&route constraints are implemented to guarantee physical independence, e.g. same value register spacing, safety coloring for power-domain routing.

Figure 8 shows an example of a design implemented with and without Functional Safety routing constraints using Innovus, the Cadence physical implementation system: the bottom-left region is the main copy of a block, while the top-right region is the replica inserted for redundancy. By guaranteeing that wires belonging to the main block can never go into the top-right region, the redundant blocks are physically independent by construction and meet requirements of the DFA.

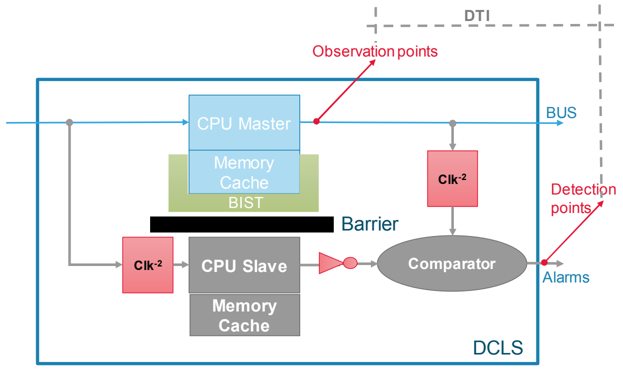


Figure 7 Dual Core Lock-Step (DCLS) architecture with countermeasures for common cause failures.

4.2 Verification

When an initial FMEDA is setup to assess the safety readiness of a IP/SoC, the DC for the safety mechanisms can be estimated based on the achievable values (low, medium, high) defined in ISO26262:2011-5, Annex D, Tables D.3 through D.9 [5].

For some standard safety mechanisms (e.g. ECC), the DC value can be calculated analytically. This computation is exact only on some parts of the logic, e.g. for ECC the DC is accurate on the data cell but not for the decoder in front of the memory. Therefore, there is a range of variability that might not be acceptable for high level ASIL targets (ASIL D) and require a more accurate investigation of the DC value by **Fault Injection**.

In the case of custom hardware or software safety mechanisms (i.e. STL, Standard Test Libraries) fault injection simulation is a technique that can be used for a more accurate verification of the DC value. It can also be used to evaluate the DTI and the Fault Reaction Time (see Figure 4) or to confirm a fault effect.

Functional safety verification is performed starting from the standard functional verification setup. In particular, a fault injection campaign to evaluate the safety mechanism DC mainly requires a description of the **workload** to execute, the **Observation Points**, i.e. where to observe the effect of faults and the **Detection Points**, i.e. where to observe the reaction of a safety mechanism.

Referring to the example in Figure 7, the observation points are placed on the primary outputs of the CPU Master, while the detection points are placed on the alarms of the comparator. The measured delay between when the fault is observed and when the fault is detected determines the DTI.

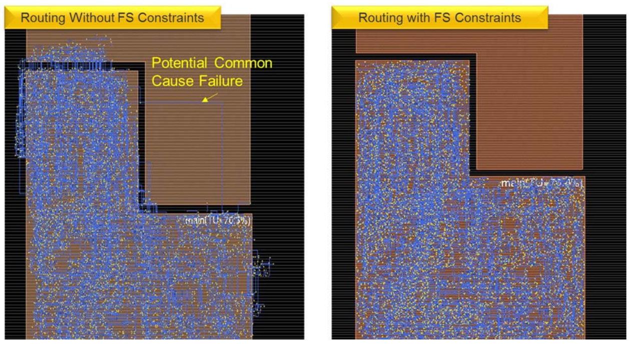

Routing Without FS Constraints
Potential Common Cause Failure
Routing with FS Constraints

Figure 8 Place and Route Functional Safety constraints to guarantee physical independence

The faults can be classified per the effects on the observation and diagnostic points:

- **Dangerous detected**: the effect of the fault is seen on both observation and diagnostic points. It means the functional output is affected by the injected fault, but the safety mechanism has detected it.
- **Dangerous undetected**: the effect on the fault is seen on the observation points, but not on the detection points. In other words, the fault affects the functional output and the safety mechanism has not detected it.
- **Safe**: the fault does not affect the observation point. It is important to note that the fault can be classified as safe only if the workload provides good coverage for functional verification.

When setting up a fault injection campaign, it is important to also decide where to inject the faults. In fact, the safety mechanism under evaluation is targeted to a specific failure mode of the circuit: fault should be injected only in the logic belonging to the related failure mode.

4.3 Software Tool Confidence Level

As part of the functional safety management to address the systematic errors, the tools used for the development of a safety critical application needs to be assessed for their confidence level. The tool confidence level (TCL) quantifies the assurance that failures in the software tools can be detected, very much along the same principles that apply to hardware components. The tool confidence level spans TCL1, TCL2 and TCL3, with TCL1 being the highest and tools classified as such do not require additional qualification and can be used in the development of applications with any ASIL. The TCL determination is based on the criteria summarized in ISO 26262-8:2011, Table 3 [5]: it consists of evaluating the tool potential impact on safety applications and the tool error detection capabilities. The information about the software tool compliance is part of the safety package developed for products to satisfy the ISO 26262 requirements for functional safety.

5 CONCLUSIONS

The race to self-driving cars and the corresponding large growth of electronics content and complexity has stretched the need for sharing the responsibility of guaranteeing functional safety through the supply chain, bridging the gap from car manufacturers/system providers to semiconductor companies and tool providers. This paper introduced the basics of functional safety and presented considerations on how design methodologies are capturing and addressing the new safety metrics through design/implementation and verification. One could envision that design methodologies will further extend to enhance support for safety requirements.

REFERENCES

[1] Maurer, Markus, et al. Autonomous driving: technical, legal and social aspects. Springer Publishing Company, Incorporated, 2016.

[2] G. Leen, D. Heffernan, “Expanding automotive electronic systems”, IEEE Computer, vol. 35 (1), 2002 pp. no.88-93.

[3] Munir, Arslan. "Safety Assessment and Design of Dependable Cybercars: For today and the future." IEEE Consumer Electronics Magazine 6.2 (2017): 69-77.

[4] IEC 61508:2007, ed.1.0 Functional Safety - Standards

[5] ISO 26262:2011 Road vehicles - Functional Safety

[6] Ismail, Azianti, and Won Jung. "Research trends in automotive functional safety." Quality, Reliability, Risk, Maintenance, and Safety Engineering (QR2MSE), 2013 International Conference on. IEEE, 2013.

[7] Beckers, Kristian, et al. "Systematic derivation of functional safety requirements for automotive systems." International Conference on Computer Safety, Reliability, and Security. Springer, Cham, 2014.

[8] Chang, Yung-Chang, et al. "Assessing automotive functional safety microprocessor with ISO 26262 hardware requirements." VLSI Design, Automation and Test (VLSI-DAT), 2014 International Symposium on. IEEE, 2014

[9] Wang, Laung-Terng, Cheng-Wen Wu, and Xiaoqing Wen. VLSI test principles and architectures: design for testability. Academic Press, 2006.

[10] Benso, Alfredo, et al. "A high-level EDA environment for the automatic insertion of HD-BIST structures." Journal of Electronic Testing 16.3 (2000): 179-184.

[11] Pateras, Steve, and Ting-Pu Tai. "Automotive semiconductor test." VLSI Design, Automation and Test (VLSI-DAT), 2017 International Symposium on. IEEE, 2017.

[12] Ruano, O., J. A. Maestro, and P. Reviriego. "A methodology for automatic insertion of selective TMR in digital circuits affected by SEUs." IEEE Transactions on Nuclear Science 56.4 (2009): 2091-2102.