Application of ISO26262 Standards on FPGA

Author: Aisha Z Azmi, Tata Elxsi & Mark Hoopes, Lattice Semiconductor

Published: November 2021

For more info: info@tataelxsi.com

Contents

ABSTRACT

FPGA adoption is increasing in industrial and automotive applications because of their high performance, low power processing, and flexibility to meet rapidly evolving requirements. Lattice Nexus™ FPGAs offer the class-leading small footprint, low power, and high reliability required for mission-critical applications. Lattice design software and tools help developers leverage Lattice FPGAs in safety-critical designs. The tools also help users easily modify or remodel logical blocks based on the reports derived by safety analysis.

While designing a safe product, a user needs to consider safety across all aspects of product development. Design challenges include adopting quality management standards, establishing a "safe" design methodology, and implementing safety concepts.

Part 11 of the ISO 26262-2018 guidelines gives FPGA designers guidance on ensuring safety in all stages of the FPGA development lifecycle. It also offers users and customers directions on how to use FPGAs in their safety applications properly.

This white paper aims to share methods to assist FPGA designers in performing Safety Analysis for FPGA-based automotive designs and offers a guide to successful ISO26262 Safety certification.

FPGA IN FUNCTIONAL SAFETY APPLICATIONS

The flexibility of FPGA provides many architectural and implementation options for developing a safe design. Furthermore, it allows the implementation of intelligent architectures, which help reduce the need for redundant design approaches used in the past, thereby reducing common cause failures.

FPGAs can be designed in such a way where only the blocks essential to achieving certification for the specific end system are used. Utilization of only fundamental blocks results in a more efficient design that only consumes the resources needed for a given Automotive Safety Integrity Level (ASIL) within the FPGA fabric.

A typical safety process for automotive applications includes developing the safety concept, determining ASIL level through risk assessment, and identifying the Safety analysis required for the target ASIL level, supported by suitable Safety management processes.

This paper will focus on the Safety Analysis procedures followed to implement a functionally safe, FPGA-based automotive design.

FUNCTIONAL SAFETY GROWTH & DESIGN CHALLENGES

The Functional Safety market was valued at USD 5,379.80 Million in 2020. It is projected to reach USD 9,418.60 Million by 2027, growing at a CAGR of 7.33 % from 2020 to 2027 (Source: Research and Markets).

FPGA-based systems have been used to create automotive designs that meet the norms of the ISO 26262 standard.

Figure 1: Functional Safety Global Market (Source: researchandmarkets)

Developing an FPGA-based automotive design that meets all the requirements of the ISO 26262 standard can pose a significant challenge. The following are some of the challenges designers must overcome to use FPGAs in safety-critical applications:

- Standard flow and templates to meet the requirements of the ISO 26262 standard
- Availability of ISO 26262-certified IP that can be used in designs
- Availability of tools that are certified to be used in ISO 26262-based designs
- Partitioning of the design to apply the hardware and software aspects of the standard
- Finding appropriate tools that are safety certified and offer special error injection and error detection methods as prescribed by the standard

The following sections describe the approaches involved to develop qualitative and quantitative Safety Analysis reports for an FPGA design.

SAFETY ANALYSIS APPROACHES

The ISO 26262 safety lifecycle encompasses the principal safety activities during the conception, development, production, operation, service, and decommissioning phases of a product.

Planning, coordinating, and documenting the safety activities of all safety life cycle phases are critical for managing a project with safety goals mandated by the ISO 26262 standard.

An example of a safety lifecycle and the safety activities involved is depicted in the figure below.

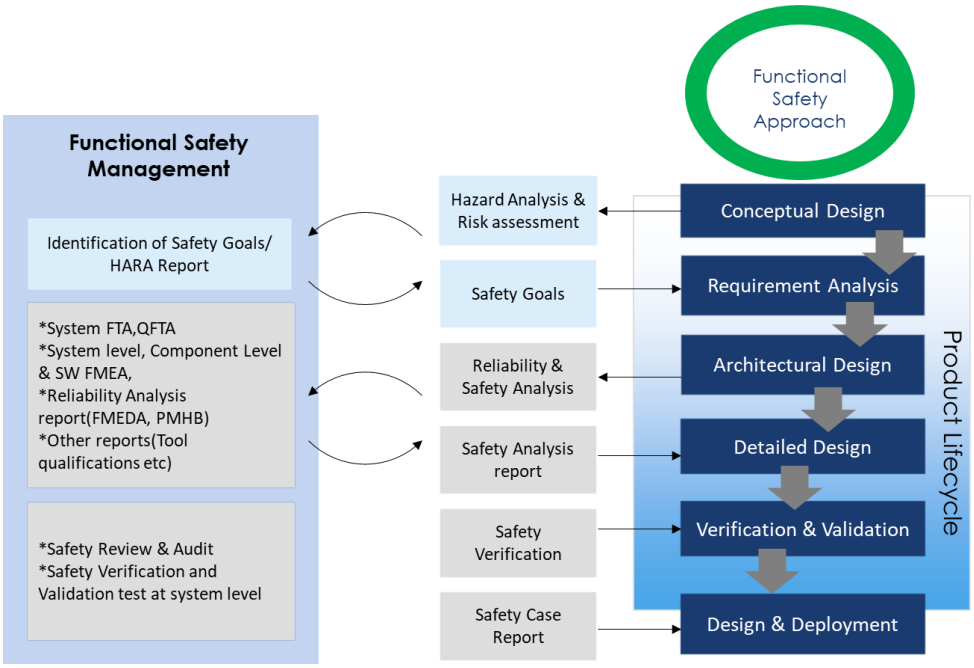

Figure 2: Safety Activities in the Overall Lifecycle

ISO 26262 Compliant FPGA Development Flow

In addition to the normal design flow, critical safety analysis steps are included at the design phase in a safety lifecycle. These analyses help by having safety mechanisms in the design and ensuring compliance with the ISO 26262 standard.

The safety analysis methods include both qualitative and quantitative approaches. This analysis, which includes parts 4, 5, and 6 of the standard (consisting of Inductive and Deductive analysis, FMEA, FMEDA, and the FTA), are described below.

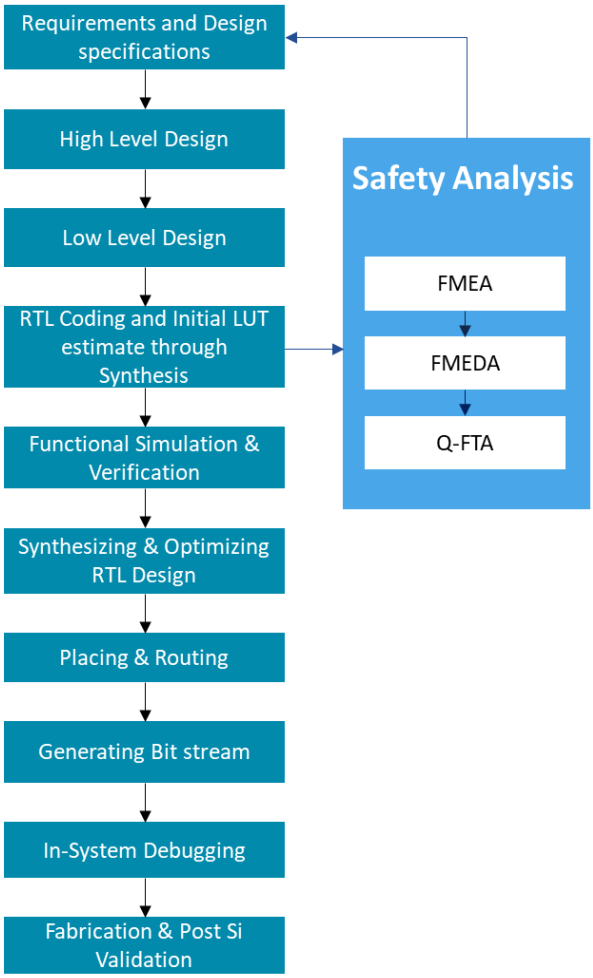

Figure 3: FPGA Design Flow

QUALITATIVE ANALYSIS

In FPGA designs targeting safety-critical applications, safety analysis is crucial to ensure there are no violations of safety goals due to a malfunction in modules. In the ISO 26262 functional safety standard, there are two methods of safety analyses - deductive and inductive. The deductive analysis is a top-down approach to safety analysis. A common method of top-down analysis is FTA - Fault Tree Analysis. Inductive analysis is a bottom-up approach of analysis, and a common methodology used is Failure Mode and Effect Analysis (FMEA).

Suppose the design includes a soft or hard CPU block. In that case, the safety life cycle must be followed for both the software and hardware development. Part 5 of the ISO 26262 standard applies to the hardware (FPGA development). Part 6 applies to embedded software development.

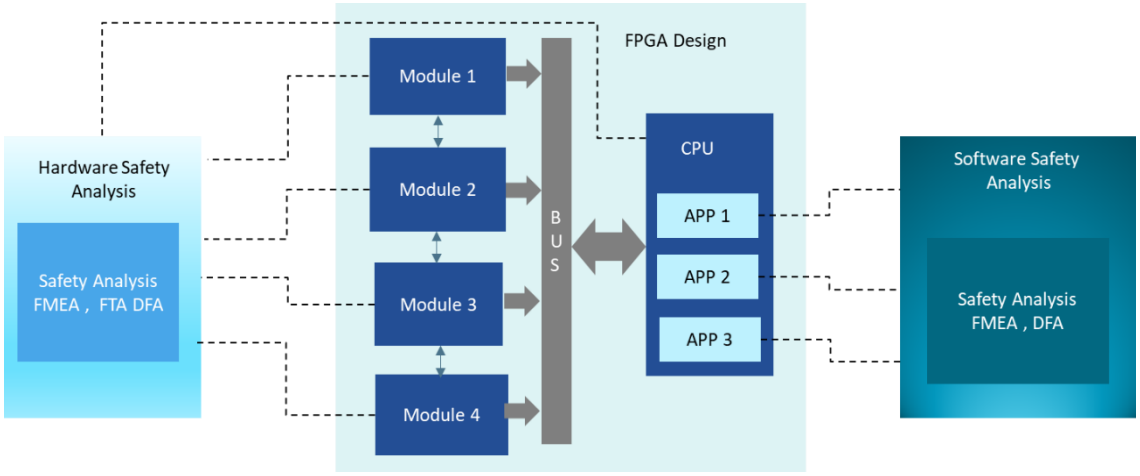

Figure 4: Split of FPGA design into Hardware and Software Parts

Failure Mode and Effect Analysis - FMEA

The Failure Mode and Effect Analysis is a qualitative safety analysis of the system to identify the potential failure modes and determine suitable safety mechanisms to address each failure.

The FPGA design architecture is broken down into simple modules based on the functionality served by each module.

Existing documentation and data are reviewed to identify the ways each module can fail.

A fault in a module can occur for various reasons. A random hardware fault such as a faulty transistor can result in the misbehavior of a module. For example, in **Error! Reference source not found.**, a random hardware failure can cause failures in the clock generation function or the reset feature.

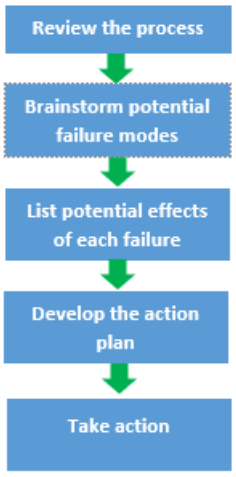

Figure 5: FMEA Flow

The effects of failures are analyzed to understand the risks at both the local and system level. Finally, safety mechanisms are set in place based on the impact of failures.

An illustration of FMEA for an FPGA UART module is shown in Table 1:

Hardware Component	Hardware Component (Part)	Hardware Sub-Component	Failure Mode	Local Effect	System Effect: (Consider if mitigation is already available)	Cause	Safety Mechanism
UART_001_1	UART Controller	rxcver.v	**Clock failure:** 1.Stuck at faults 2.DC model faults	No UART transaction	Loss of UART RX data	Random HW fault	SM_024
UART_001_2			**Reset failure:** 1.Stuck at faults (continous reset) 2.DC model faults	No UART transaction	Loss of UART RX data	Random HW fault	SM_024
UART_001_2_1			**Reset failure:** 1.Stuck at faults (no reset)	No reset of UART registers	Corruption or loss of UART data	Random HW fault	SM_024, SM_036, SM_037
			Control register corruption				
UART_001_3			**Hunt bit corruption:** Stuck at faults, False bit set	It will recognize false start of frame and receive incorrect data.	Corrupted UART data reception	Random HW fault	SM_037
UART_001_4			**Hunt one bit corruption:** Stuck at faults, False bit set	No intimation about the incorrect data received will be intimated	Corrupted UART Data Reception	Random HW fault	SM_037
UART_001_5			**rbr_datardy bit corruption:** Stuck at faults, False bit set	Stuck at 1: CPU will continuously get false indication that new frame has come. Stuck at 0: CPU will not know new frame as come and will loose frames.	Corrupted UART Data Reception	Random HW fault	SM_037

Table 1: FMEA of an FPGA module

Upon enumeration of all the failures affecting the local and system level, the design is analyzed further to see if safety mechanisms are in place to ensure detection and recovery from the faults. In an FPGA-based design, the safety mechanisms can be implemented as intelligent error detection and correction methods.

A similar FMEA is also done for the software components in the system. Each software function is analyzed to identify failure modes and create safety mechanisms/diagnostics to detect those failures. It enables the system to transition to a safe state in case of any occurrence fault.

When third-party components are used in the system, the designers' responsibility is to ensure that all component features utilized are analyzed for safety violations and safety mechanisms/diagnostics are available at the system level in case of failures.

Based on the improvements identified by the Process Failure Mode and Effects Analysis team, the FPGA architecture design can be re-evaluated.

Each potential failure is re-evaluated. Once the improvements have been made, the architecture can be refined and re-analyzed repeatedly to reduce the severity of inevitable failures.

Due to their re-programmability, long lifespans, and high processing bandwidth, FPGA architectures can be tailored to suit the safety needs at various stages of an automotive project.

Safety Mechanisms for Random Failures

- Redundancy
  - Triple modular redundancy (TMR)
  - Duplication with Compare (DWC)

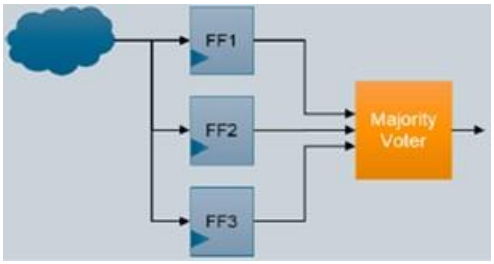

Figure 6: Triple Modular Redundancy

TMR uses a voting mechanism to identify which module has failed and to enable error detection and recovery. However, this method has the side effect of increased gate count.

A duplication with compare scheme (DWC) may be sufficient in some cases, which can help in fault detection (but not in recovery from the fault) Fig 7.

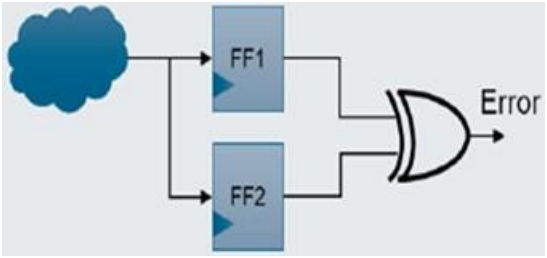

Figure 7: Duplication with Compare

A safety mechanism based on redundancy can be achieved by providing CPUs operating in Dual Core Lock Step mode (Fig 8).

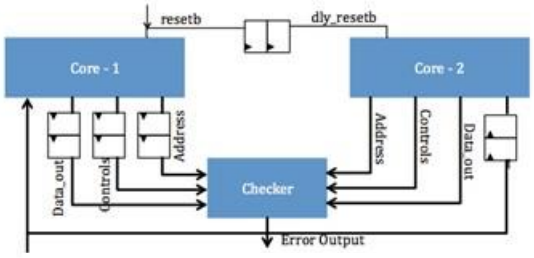

Figure 8: Dual Core Lock Step

- Error Correction methods

  These methods can ensure the system can continue running in the event of a failure by entering a Safe state. e.g., SEC: Single-bit error correction using ECC for memory corruption in Embedded RAMs.

- Diagnostics and Error detection methods

  Error detection logic can be implemented in hardware to inform the system controller of failures for further action to transition to a safe state. For example,

- Detection of Data Integrity errors
- Errors in clock generation logic
- Errors in the timing of interface signals
- 2-bit memory error detection via ECC

These errors can be indicated by setting error bits in status registers, resulting in resetting the whole system or the host discarding the error packets.

Error detection/diagnostics can also be implemented in software. For example,

- Regular checking of important control registers
- Running diagnostics on external devices like onboard sensors or physical layer components (like Ethernet PHY devices)
- CRC validation of software and configuration partitions on boot

In the case of such diagnostics, especially those implemented in software, the designer must ensure that system-level parameters like the fault-tolerant time interval (FTTI) are met.

Systematic Failures

Systematic faults may be caused during design, development, and manufacturing due to errors in specification or implementation.

- Quality Control
  - This may include the removal of the potential failure through testing or inspection. The inspection effectiveness must match the level of severity that the hazard may impose on the consumer.
  - Various validation and error injection tools can be used for adequate testing, such as HDL Simulators, Fault Simulators, and Static Code Analysis tools. The process for tool selection is described in the sections below.

Fault Tree Analysis - FTA

The other qualitative method of safety analysis is the Fault tree Analysis (FTA). FTA is deductive and involves a logical breakdown from the Top-level undesired event (Safety goal violation), cascaded to the Base-level event (root cause).

The Inductive and the Deductive approaches are complementary as stated in ISO 26262-5: 2018 7.4.3.1, Table 2 NOTE: “The level of detail of the analysis is commensurate with the level of detail of the design. In certain cases, both methods can be carried out at different levels of detail.”

The fault tree construction is done by adding the Top Event, the Safety Goal, or Safety requirement considered for the analysis.

The top event is then analyzed to split into potential failure events. The hardware architecture is taken as an input for this. Finally, a root cause analysis of failures that could lead to the failure of Top Event is performed.

As shown in Figure 9, the base events are added to the abstraction level of system elements where FTA is drawn at a system level.

At hardware level FTA, these base events will be expanded to the lowest level of abstraction in hardware or up to hardware module level as the Safety design requires.

To derive the quantitative scores of the probability of each Base level failure event, the fault rate of each event is assigned.
This is described in the sections below.

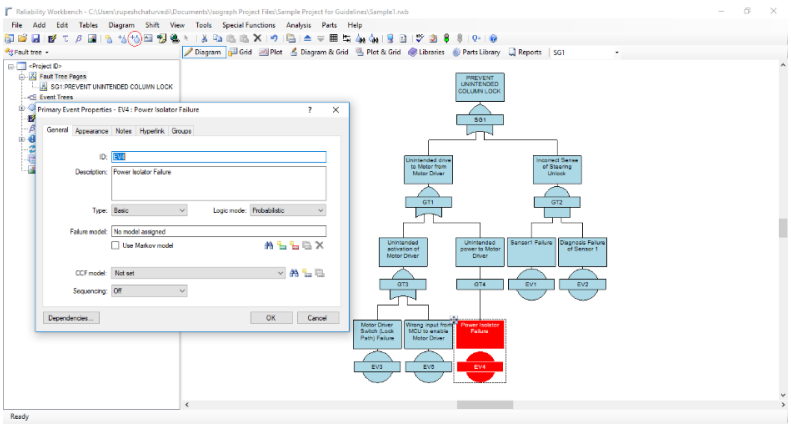

Figure 9: Fault Tree in the Isograph Tool

A typical approach (Per ISO 26262:10, B.2) at item level is to use the FTA to analyze the violation of safety goals and identify hazardous events down to the component level. Failure modes of the hardware modules are then analyzed from bottom-up analysis using FMEA. This way, FTA and FMEA can be combined to balance top-down and bottom-up safety analysis at the item level (Fig 10).

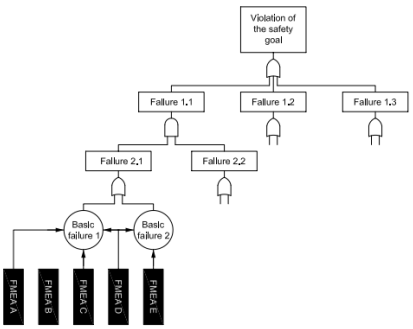

Figure 10: Illustration of a combination of FTA and FMEA

Dependant Failure Analysis - DFA

Dependent Failure analysis (DFA) is also carried out to mitigate the Safety risks occurring from common causes and cascading failures.

Examples of dependent failures due to shared resources' random hardware faults are clock elements, power supply elements, or common reset logic. Examples of Dependent failures associated with random physical root causes include short circuits, latch-up, and crosstalk.

Typical countermeasures for dependant failures include:

- 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 and 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)

QUANTITATIVE ANALYSIS

Quantitative analysis is carried out to derive the Single point, Latent and Probabilistic Fault metrics of a system. The Failure Mode Effect Diagnostic analysis (FMEDA) is a quantitative analysis used to derive the Single Point and Latent Fault metric of a system. FMEDA helps in identifying failure modes and the probability of occurrence. In addition, it helps in assessing and improving the failure diagnostic capability of a system.

It is used to calculate the hardware architectural metrics such as the Single point fault metric (SPFM) and latent fault metric (LFM) of the hardware. These metrics can then be used to derive the probabilistic metric of random hardware failures (PMHF).

For the development of FMEDA, the FMEA analysis is taken as the base. Thus, the first step for FMEDA would be to derive the failure rate of each module.

For an FPGA-based design, the number of LUT and Block RAM used in a module can be used to derive the Failure rate of each module. The FPGA design synthesis tool can be used to derive the LUT and Block RAM count per module.

In the snapshot in **Error! Reference source not found.**11, different modules of the initial draft of RTL design are listed, and a count of LUTs and Block RAMs are derived per module.

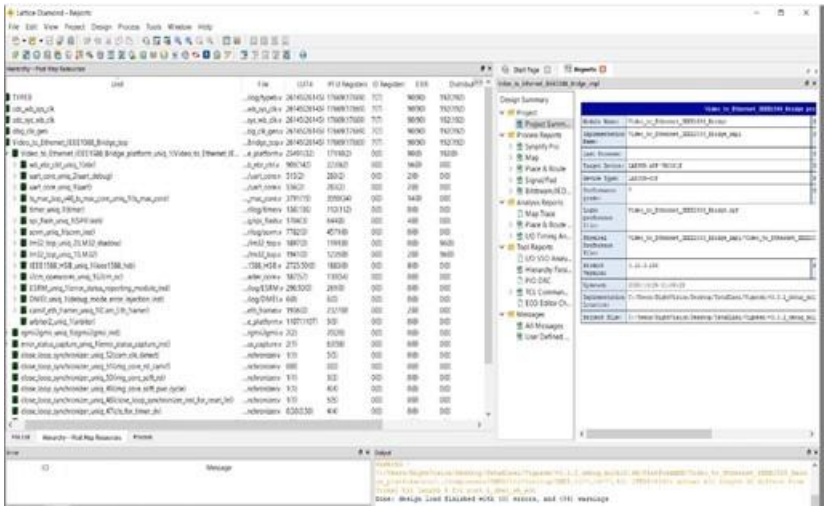

Figure 11: Lattice Diamond Tool for Synthesis of RTL Design to get LUT count

The failure rate (FIT) of each LUT & Block RAM is calculated based on the mathematical equation shown below:

$$\lambda_{die} = \{\lambda_1 \times N \times e^{-0.35\times\alpha} + \lambda_2\} \times \left\{\frac{\sum_{i=1}^{y}(\pi_t)_i \times \tau_i}{\tau_{on} + \tau_{off}}\right\}$$

$$\lambda_{package} = 2.75 \times 10^{-3} \times \pi_a \times \left(\sum_{i=1}^{z}(\pi_n)_i \times (\Delta T_i)^{0.68}\right) \times \lambda_3$$

$$\lambda_{overstress} = \pi_I \times \lambda_{EOS}$$

$$\lambda = \{\lambda_{die} + \lambda_{package} + \lambda_{overstress}\} \times \frac{10^{-9}}{h}$$

Equation 1: Mathematical model for reliability prediction

Based on the above λ values obtained from the standard and the Mathematical equation in Equation 1, the Failure rate per LUT and Block RAM is calculated for the FPGA. An example of all the contributing factors is shown in Table 2. The FIT/unit is then obtained.

Die Failure Rate												
Element	Transistors / unit	Total Count	λ1	N	α (Year of Manufacturing = 2020)	λ2	$\lambda 1_{eff}$	Base FIT	De-rating for temp	**Effective FIT**	**FIT/Unit**	
LUTs(#)	100	44000	0.00002	4400000	22	34.00	0.03984879	34.0398	0.08253310	**2.80941435**	**0.00047100**	per LUT
EBR(bits)	6	1990656	0.00000017	11943936	22	8.80	0.00091945	8.8009	0.08253310	**0.72636719**	**0.45700111**	per EBR
Package Failure Rate												
Package Type	ΔT_j (°C)	S (Number of Pins)	D (mm)	π_a	λ3 (FIT)	De-rating for temp	Effective FIT	λ_{pin} **(FIT)**	**FIT/transistor**	**% of package FIT for Logic**	**FIT/LUT**	**FIT/EBR**
CABGA 381	26.27	381	24.04163056	0.33	15.25254252	6009.418	83.18044878	**0.17465711**	**4.072E-06**	**80.00%**	0.00040715	0.450275488
Electrical Overstress												
λ_{EOS}												
20												

Table 1: Example of Die & Package Failure rate of a Lattice FPGA

After determining failure rates and failure modes for each module, the next important step is to achieve the diagnostic coverage of each module based on the safety mechanisms assigned for each failure. Again, annex D of Part 5 in the standard can be used to determine the scores of diagnostic coverage assumed for each safety mechanism.

Hardware Component ID	Hardware Component (Part)	HardwareSub-Component	Describe the HW function (Sub-part)	Failure Mode	LUTs required	EBR	Cause	Safety Mechanism ID (Single-Point)	Failure distribution	$\lambda_{SR} = \lambda$ [FIT]	Fsafe, 1 if safe, 0 if not safe (and Fsafeavg per cmpnt)	λS failure mode	λS component	λnS failure mode	F_{PVSG}, 1 if potential, 0 if no potential for PVSG	λPVSG failure mode	$\lambda_{SPF,ar}$ (and $\lambda_{MPF,ar}$ per cmpnt)
I2C_001_2	I2C Controller	i2c_master_core.v	Start Signal Generation:	No start signal generation	61	0	Random HW fault	Watchdog Timer	0.04	**0.083053476**	0.5	0.0018	0.0054	0.0018056	1	0.0018056	0.99

Table D.10 — Program sequence monitoring/Clock

Safety mechanism/measure	See overview of techniques	Typical diagnostic coverage considered achievable	Notes
Watchdog with separate time base without time-window	D.2.9.1	Low	—
Watchdog with separate time base and time-window	D.2.9.2	Medium	Depends on time restriction for the time-window
Logical monitoring of program sequence	D.2.9.3	Medium	Only effective against clock failures if external temporal events influence the logical program flow. Provides coverage for internal hardware failures (such as interrupt frequency errors) that can cause the software to run out of sequence
Combination of temporal and logical monitoring of program sequence	D.2.9.4	High	—
Combination of temporal and logical monitoring of program sequences with time dependency	D.2.9.5	High	Provides coverage for internal hardware failures that can cause the software to run out of sequence. When implemented with asymmetrical designs, provides coverage regarding communication sequence between main and monitoring device NOTE Method to be designed to account for execution jitter from interrupts, CPU loading, etc.

Table 2: Example of an FPGA Module with the diagnostic coverage

The example in Table 3 shows an I2C module in an FPGA. The number of LUTs associated with the clock generation function gives us an estimate of the failure rate associated with the subcomponent. The failure rate is then split into safe and non-safe failures. For a logical design, 50 percent of the failures are considered unsafe. (Per ISO 26262:10, 8.1.8 - g)

A diagnostic coverage is then associated with the failure based on the safety mechanism chosen. For example, in the diagram below, the safety mechanism selected for the clock generation function is the watchdog timer. This can be considered as a combination of temporal and logical monitoring of the program sequence. Therefore, the DC associated with it is high (99 percent). A revised safety mechanism can be used in case the diagnostic coverage is low.

FMEDA provides information about the failure mode that can be identified and connected to appropriate FTA events. Thus, correlating FMEDA events to FTA events can achieve complet e safety analysis at the item level.

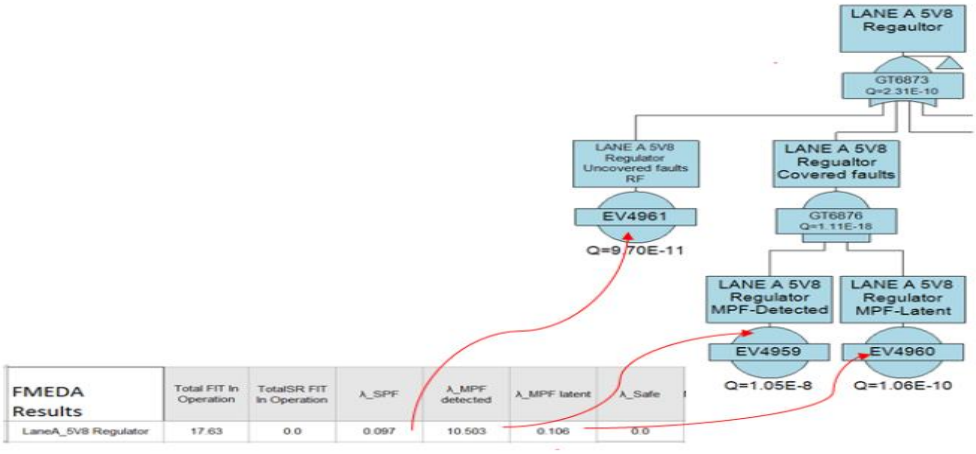

Figure 12: Failures Rates in Quantitative FTA derived from FMEDA

The quantitative FTA is used to derive the PMHF of the system. For the quantitative FTA, the qualitative FTA is taken as the base and further events/gates are added to base events in Hardware FTA. Once the top-level faults are analyzed to the base event at the hardware level, the failure rate is added to the base event.

The failure rates can be derived from either the Prediction libraries such as the MIL-HDBK-217 Military Handbook or from the FMEDA. This example shows the result of the 5V8 regulator module from FMEDA, used to feed the base events in the prescribed pattern. The same can be repeated for each of the identified hardware sub-modules leading to a violation of the safety goal in hardware FTA. The failure rates of the base events are added for all safety-related hardware parts that have the potential to violate the safety goal to the relevant Gates. The sum of failure rates of all modules provides the final score. Once complete the PMHF for the fault tree can be derived.

Part 10 of the ISO26262-2011 gives a detailed explanation of the factors associated with PMHF calculation. The formula used for PMHF calculation includes the combined failure rates of Residual and Multipoint failures obtained from the FMEDA.

$$M_{\text{PMHF}} = \lambda_{\text{RF}} + \lambda_{\text{m,DPF}} \times \lambda_{\text{sm,DPF,latent}} \times T_{\text{Lifetime}}$$

Equation 2: Equation for PMFH calculation

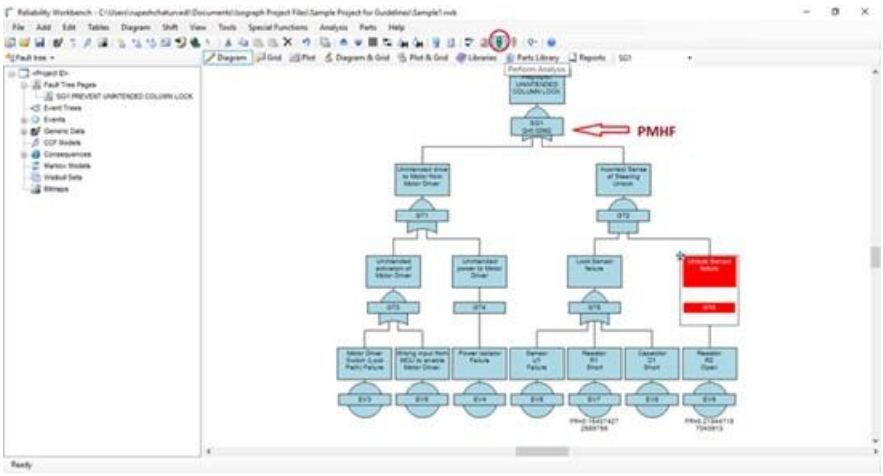

Figure 13: PMHF derived from Q-FTA in Isograph Tool

Timing

One area important to all aspects of Functional Safety design is how much time is available to recognize that something has failed and then to take action to prevent a hazard from occurring. This period is referred to as the Fault Tolerant Time Interval (FTTI). To ensure the safety of a system, the time from the detection of a fault plus the time for the system to achieve a safe state shall be less than the FTTI for the safety goal.

Diagnostic Test Interval (DTI) + Fault Reaction Time Interval (FRTI) < FTTI

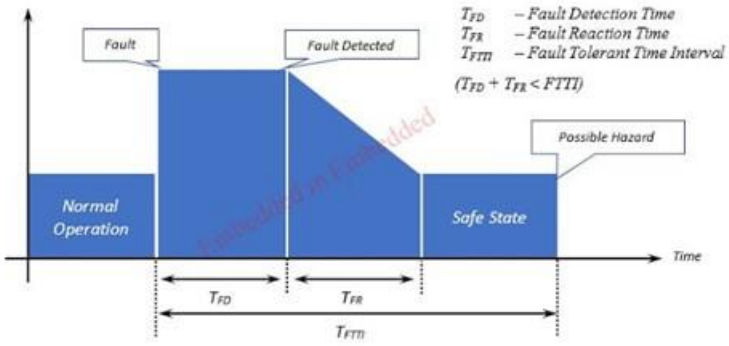

Figure 14: Fault Tolerant Time Interval

SUPPORTING PROCESSES

Apart from the Safety management processes, the other critical aspects of a Safety project involve thorough reviews, traceability of work products, and qualification of the third-party IPs and tools used.

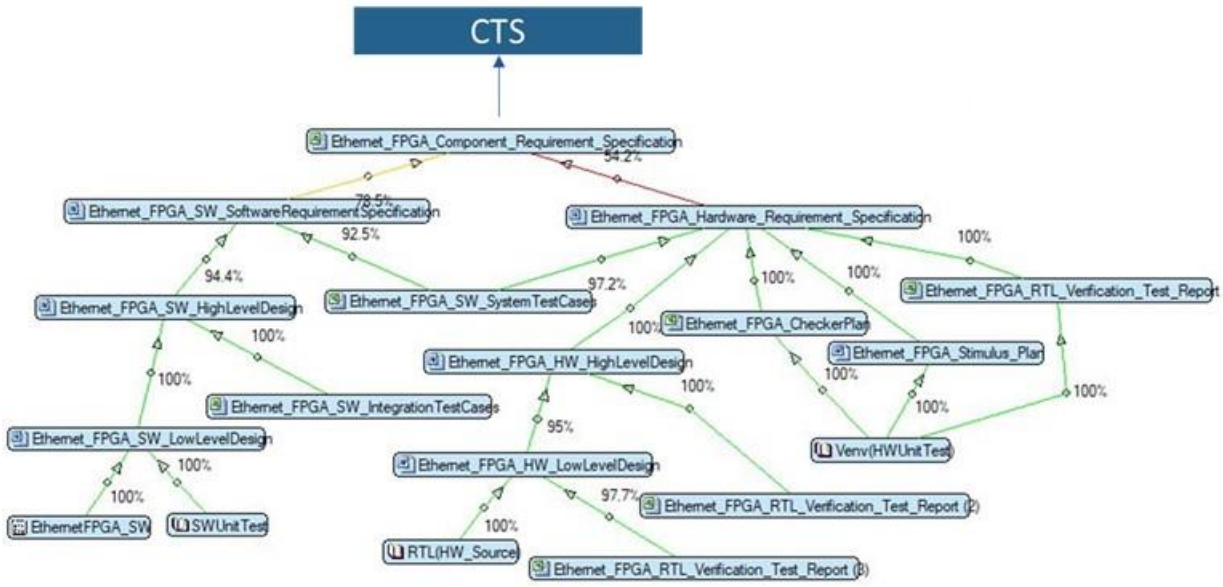

Figure 15: Traceability Map using Reqtify Tool

While reviews may be manual, various levels of reviews such as confirmation reviews, inspection reviews, and safety audits can ensure the prevention of any Systemic failures occurring in the project.

For ensuring traceability of work products at all stages of the project, traceability tools such as Reqtify or Doors can be used.

The traceability tools ensure that all identified safety features are tracked down to suitable functions within the FPGA, keeping in sync with the identified safety lifecycle.

Figure 15 is an example of the traceability map obtained from the Reqtify tool. This map shows the requirement specification split into hardware and software requirements, the associated High level and Low-level design documents, and the Test plans associated with each part.

The reliance on tools for ensuring a Safe design gives rise to the need of qualifying the tools to be safe. In a safety-critical project, tool qualification is only needed for some tools. For determining if a tool qualification is required or not, the ISO 26262 (ISO 26262-8:2011, Table 3) describes a simple two-step process to find out how much confidence we need to have in a particular tool. If no confidence is needed, we get the lowest tool confidence level (TCL 1) and a tool qualification is not needed. Test tools (like the Polyspace Static Analysis tool, Unity Unit testing tool, simulators, etc.) usually require a high level of confidence and, therefore a tool qualification is a must.

While tool qualification can be an expensive and time-consuming process, a tool pre-qualification from authorities like TÜV can almost eliminate the qualification efforts on the user side. The tool vendors may also provide steps to qualify a tool or safety certification of the tools used. The Lattice Diamond® 3.10 SP3 (Build 3.10.3.144.3) software tools and tool flow satisfy the requirements for "Validation of the software tool" up to TCL3 for ASIL D. Furthermore, the Lattice device libraries with their library elements satisfy the requirements for Safety-Element-Out-Of-Context (SEooC) for ASIL D.

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.

Also, in the course of a project, various third part IPs may be procured and used to ensure a quicker and well-established architecture. These third-party IPs also need to be qualified to ensure they are safe to use. Module-level tests, FMEA, and DFA analysis can ensure the safety of using these IPs.

CONCLUSION

The competitive landscape requires manufacturers to develop intelligent, functionally safe designs that provide quality products while meeting competitive time-to-market, cost, and performance targets. Tata Elxsi as a design partner has successfully applied the different parts of ISO 26262 standard across all modules of a design based on a Lattice FPGA to achieve a safe design with simple application processes and a quick turnaround time. In addition, Tata Elxsi has also shown effective ways for tool usage and developing safety literature during the ongoing safety analysis.

FPGA-based design methodology changes the implementation for safety designs and greatly reduces development effort, system complexity, and time to market. The safety approaches described above include both Inductive and Deductive safety analysis to realize a safe design, and the users can follow a combination of both to achieve the desired results.

The safety approaches allow FPGA users to design tailored safety systems and controllers and provide a substantial competitive advantage over traditional microcontroller or ASIC-based designs.

ABOUT TATA ELXSI

Tata Elxsi is recognized as a premium engineering and design service provider worldwide and amongst the leaders in the automotive, media, broadcast, communications, and healthcare industries for system design and development. Tata Elxsi's integrated Design and Technology teams help enterprises reimagine their products and services – from strategy, consumer research and insights, to service and experience design, technology implementation, integration, launch, and beyond.

ABOUT Lattice Semiconductor

Lattice Semiconductor is the low-power programmable leader. We solve customer problems across the network, from the Edge to the Cloud, in the growing communications, computing, industrial, automotive, and consumer markets. Our technology, long-standing relationships, and commitment to world-class support lets our customers quickly and easily unleash their innovation to create a smart, secure and connected world.

REFERENCES

1. ISO 26262 standard publication - ISO 26262 (2011) and ISO 26262 (2018)
2. Functional safety packages from Lattice Semiconductor
3. FMEA training resources
4. Basics of FMEDA and how it is helpful in system-level safety analysis
5. Fault Tree Analysis, Quality-one
6. When and how to qualify tools according to ISO 26262 by Markus Gros, 2019
7. Functional safety market size and forecast by Verified Market Research, 2021
8. Global Functional safety market by Research And Markets, 2020

